

**USER'S GUIDE FOR AN AIRBORNE DOPPLER
WEATHER RADAR SIMULATION
(ADWRS)**

(Version 5.1)

INFORMAL REPORT

**NASA Contract NAS1-99074
Task Order No. 1029**

Prepared for

**National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23681**

**USER'S GUIDE FOR AN AIRBORNE DOPPLER
WEATHER RADAR SIMULATION
(ADWRS)**

(Version 5.1)

By
Charles L. Britt, Ph. D.
Carol W. Kelly

RTI
Center for Aerospace Technology

INFORMAL REPORT

**NASA Contract NAS1-99074
Task Order No. 1029**

Prepared for

**National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23681**

TABLE OF CONTENTS

TABLE OF CONTENTS	ii
LIST OF FIGURES.....	iii
LIST OF TABLES.....	iii
1.0 INTRODUCTION	1
2.0 GENERAL DESCRIPTION	2
2.1 Weather Models.....	4
2.2 Clutter Model.....	5
2.3 Discrete Target Model	6
3.0 PROGRAM STRUCTURE	8
3.1 Main Program	11
3.2 Subroutines	11
4.0 INPUT FILES.....	14
4.1 Parameter Definition File.....	14
4.2 Filename Definition File.....	19
4.3 Weather Data Input Files (3D).....	20
4.4 Antenna Pattern Data File.....	21
5.0 GRAPHICAL USER INTERFACE (GUI).....	23
6.0 OUTPUT FILES	26
6.1 Post Processing Data Output File	26
6.2 Processed Data Output File — Comma Separated Values	28
6.3 I and Q Data Output File.....	30
6.4 Doppler Velocity Spectra Output File	32
APPENDIX A.....	A-1
APPENDIX B.....	B-1
APPENDIX C	C-1

LIST OF FIGURES

Figure 1. General Operation of ADWRS.....	3
Figure 2. Analytical/Empirical Ground Clutter Model in ADWRS.....	5
Figure 3. Logical Flow Diagram Showing the Main Calculation Loops in ADWRS.....	8
Figure 4. Breakdown of a Range Bin into Incremental Scattering Cells.....	9
Figure 5. Techniques for Calculation of In-phase (I) and Quadrature (Q) Signals.....	10
Figure 6. Filename Definition File	20
Figure 7. ADWRS GUI Main Menu.....	24
Figure 8. Radar Parameter Submenu	24
Figure 9. Simulation Parameter Submenu	25
Figure 10. Aircraft Parameter Submenu	25

LIST OF TABLES

Table I. Values of Empirical Constants at 10 GHz for Analytical Ground Clutter Model	5
Table II. Functions of Major Subroutines in ADWRS	11
Table III. Description of Parameter Definition Input File.....	15
Table IV. File Header for Post Processing Data Output File.....	26
Table V. Line Header for Post Processing Data Output File.....	27
Table VI. Bin header for Post Processing Data Output File.....	28
Table VII. Data for Post Processing Data Output File	28
Table VIII. Contents of Processed Data Output File.....	29
Table IX. Headers for Processed Data Output File	29
Table X. Headers and Data Contents for I & Q Data Output File.....	30
Table XI. Contents for Spectral Data Output File	32

1.0 INTRODUCTION

This report provides a description of the Airborne Doppler Wind Shear Radar Simulation (ADWRS) developed for NASA-Langley Research Center by RTI. The simulation provides a comprehensive calculation of the signal characteristics and expected outputs of an airborne, coherent, pulsed Doppler radar system. Early versions of the simulation were used in the Wind shear Radar Program. Those capabilities have been extended for use by the NASA Turbulence Program.

The simulation contains algorithms for direct calculation of radar signal returns from data files providing wind velocities and the radar reflectivity of moisture at any point in space, combined with a calculation of the scattering from ground clutter and discrete targets. The instantaneous radar signal amplitude is calculated by spatially integrating the radar equation over a large population of incremental scatterers. Additional calculations simulate the signal processing done by a coherent radar in filtering the signal, providing Automatic Gain Control (AGC), forming in-phase (I) and quadrature (Q) base-band signal components, converting the I and Q signals to digital values, filtering the I and Q signals, and deriving Doppler velocity, spectral width, shear hazard and other radar outputs of interest.

The detailed nature of the simulation permits the evaluation of proposed trade-offs in radar system parameters and the evaluation of the performance of proposed configurations in various weather/clutter environments. The simulation also provides a test bed for proposed signal processing techniques for minimizing the effects of noise, phase jitter, and ground clutter and maximizing the useful information derived for avoidance of turbulence or microburst wind shear by aircraft.

2.0 GENERAL DESCRIPTION

Figure 1 contains a block diagram indicating the major features of the simulation. Inputs to the program include an input file specifying the radar system parameters and data files that describe the characteristics of the weather. The weather data files provide reflectivity factors and x-y-z components of wind velocity. The weather database can be generated by various models, including the Frehlich/NCAR 3-component, 3-D von Karman turbulence model, the AeroTech/NCAR interpolated model with superimposed von Karman turbulence, the NASA TASS model [2], and the NCAR cloud physics model. The outputs of these models must be placed in a file format compatible with ADWRS using supplementary programs. Other inputs specify the position and velocity of the simulated airborne radar and the radar scanning strategy.

The simulation starts at the initial position of the aircraft and the initial scan position of the radar antenna. As the simulation proceeds, the radar antenna moves through the specified raster scan range. After an azimuth scan is completed, the position of the aircraft is updated, antenna tilt is changed (if specified), and the next azimuth scan is started in a reverse direction. This process continues for the number of scans and aircraft positions specified in the input file.

For each range bin along a radar line-of-sight, the simulation calculates the level of the received signal amplitude by integrating the product of the antenna gain pattern and scattering source amplitude and phase over a spherical-shell volume segment defined by the pulse width, radar range and ground plane intersection. The amplitude of the return from each incremental scatterer in the volume segment is proportional to either the square root of the normalized cross-section of the ground clutter or the square root of the reflectivity factor of the moisture in the weather. The phase of each incremental scatterer is the sum of: 1) a uniformly distributed ($0 - 2\pi$) random phase term, 2) a phase term due to relative aircraft-scatterer radial velocity, and 3) normally distributed random phase terms representing transmitter/receiver phase jitter and ground clutter random motion. The random phase terms simulating phase jitter and ground clutter motion are updated for each transmitted pulse, while the uniformly distributed phase terms are updated for each sequence of pulses in a range bin. The phase terms representing aircraft-scatterer relative motion are linear functions of time.

Path attenuation for each incremental scatterer is determined by integrating the path losses over the transmission path. Empirical formulas from reference [1] are used to determine

the incremental path losses from the moisture content of the turbulence or microburst. The simulation provides the option to subtract aircraft ground velocity so that derived Doppler frequencies can be referenced to a value of zero ground speed, if desired. Simulated antenna patterns include a generic parabolic antenna with size and aperture illumination taper specified by input data and a flat-plate array antenna with a pattern similar to that found in the current generation of X-band airborne weather radars.

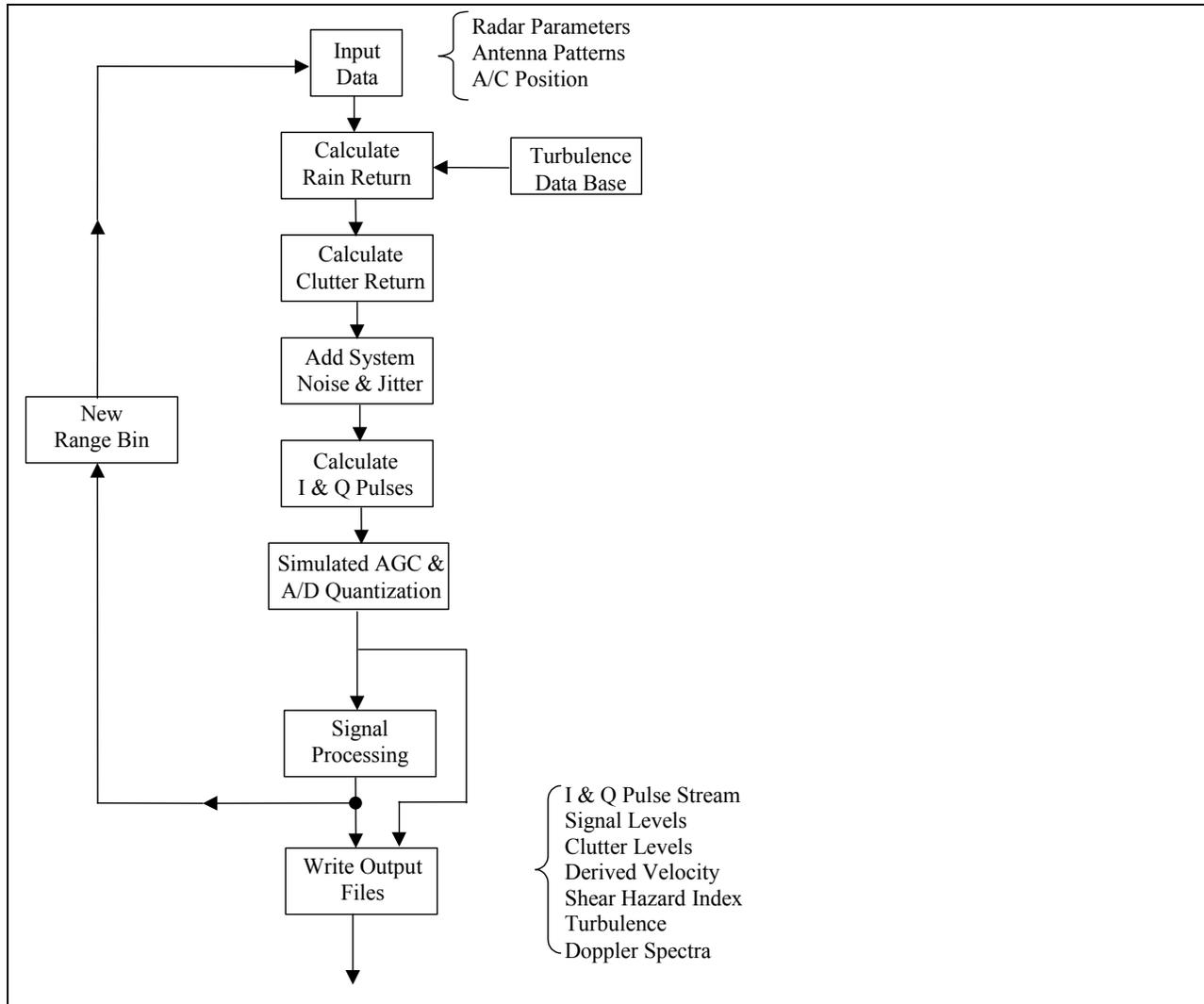


Figure 1. General Operation of ADWRS.

A sequence of N pulses of in-phase (I) and quadrature (Q) signal amplitudes are calculated for each range bin, as discussed above, and written to an output file. In subsequent processing, these I and Q signal amplitudes are subjected to automatic gain control (AGC) amplification and analog-to-digital (A/D) quantization. A simulated, fast-acting AGC is used to

adjust the gain of the system on a bin-by-bin basis to achieve a wide dynamic range and to prevent signal saturation (due to clutter) prior to and during A/D conversion.

For wind shear processing, the I and Q pulse stream is digitally filtered to suppress ground clutter near zero Doppler frequencies and processed using both conventional pulse-pair and spectral averaging algorithms to derive the average velocity and spectral width of the scatterers in the range bin. Additional processing of the velocity data provides information on wind shear and aircraft hazard factor. It should be noted that the ground clutter filters are time-domain filters, and only apply if the Doppler effect due to relative aircraft/ground velocity along the LOS is removed from the I and Q data prior to filtering. In the NASA experimental wind shear radar, the aircraft Doppler term was removed from the IQ signals by adjusting the frequency of the local oscillator (LO) in the radar system. In the NASA experimental turbulence detection radar, this method is not used and the LO frequency is fixed. In ADWRS versions 5.0 or greater, an option is provided to either remove the effect of aircraft velocity or not.

As mentioned previously, provision is made in the simulation to generate returns from a specified number of range bins over a specified azimuth scan so that color displays of reflectivity, velocity, shear, spectral width, etc., can be created and examined using other applications. Other outputs of the simulation include plots of power levels, velocity, spectral width, shear hazard factor, AGC levels vs. radar range, and Doppler spectra of ground clutter and moisture, as derived from the I and Q signals from each simulated range bin.

2.1 Weather Models

ADWRS accepts weather data input from the virtually any weather model, providing the outputs of the model is placed in a file format compatible with ADWRS, using supplementary programs if necessary. The resulting weather data files must provide reflectivity factors and x-y-z components of wind velocity. ADWRS is intended for use with the following models: the Frehlich/NCAR 3-component, 3-D von Karman turbulence model, the AeroTech/NCAR interpolated model with superimposed von Karman turbulence, the NASA TASS model, and the NCAR cloud physics model.

2.2 Clutter Model

Ground clutter is calculated from an empirical model that removes the reliance of previous versions of ADWRS on SAR clutter maps of the Denver area. Moving clutter from discrete targets is implemented in version 5.1 of ADWRS as an option.

An analytical/empirical model for ground clutter is given in reference [4]. This model is based on fitting curves to ground clutter data obtained from many different clutter measurement programs. See Figure 2.

$$\sigma^0 = A(\theta + C)^B \exp\left[\frac{-D}{(1 + K)}\right]$$

$$\text{where } K = \frac{0.1\sigma_h}{\lambda}$$

A, B, C, and D \equiv empirically determined constants depending on the type of surface viewed

σ_h \equiv the standard deviation of the surface height in units consistent with the wavelength λ

θ \equiv the depression angle in radians

Figure 2. Analytical/Empirical Ground Clutter Model in ADWRS.

Values for the constants for various surfaces at X-band (10 GHz) are listed in Table I. These are the values used in ADWRS. Note that the standard deviation of the surface height is only used when the surface is soil. The program will need to be modified if use of other frequencies is desired. Values of the constants for other radar frequencies are given in reference [4].

Table I. Values of Empirical Constants at 10 GHz for Analytical Ground Clutter Model

Constant	Soil	Grass	Tall Grass	Trees	Urban	Wet Snow	Dry Snow
A	0.25	0.023	0.006	0.002	2.0	0.025	0.195
B	0.83	1.5	1.5	0.64	1.8	1.7	1.7
C	0.0013	0.012	0.012	0.002	0.015	0.0016	0.0016
D	2.3	0.0	0.0	0.0	0.0	0.0	0.0

2.3 Discrete Target Model

Provision is made in ADWRS to optionally simulate stationary and moving discrete targets such as ground objects, other aircraft, automobiles, etc. To specify the characteristics of these targets, the discrete target file is used. This file provides the coordinates of the targets, the velocity components, and the radar cross-sections of the targets. The targets are organized into segments consisting of up to 80 segments with up to 1000 targets per segment. In previous simulations of wind shear, a series of segments were used to represent automobiles along an interstate highway.

The file data is organized as ASCII numbers as follows:

Line1: Number of segments (N)

Line2: Number of targets in segment 1, number of targets in segment 2, etc.

Line3: X coordinate values for targets in segment 1, X coordinate values for targets in segment 2, etc., for all segments N (meters)

Line 4: Y coordinate values for segment 1, Y coordinate values for segment 2, etc., for all segments N (meters)

Line 5: Altitude coordinate values for targets in segment 1, Altitude coordinate values for targets in segment 2, etc., for all segments N (meters)

Line 6: RCS values for targets in segment 1, RCS values for targets in segment 2, etc., for all segments N (square meters)

Line7: X coordinate velocity for targets in segment 1, X coordinate velocity for targets in segment 2, etc., for all segments N (meters/second)

Line8: Y coordinate velocity for targets in segment 1, Y coordinate velocity for targets in segment 2, etc., for all segments N (meters/second)

A FORTRAN listing for the subroutine for reading the data is shown below.

```
NAME: RDDISC2()
C
C  PURPOSE: VERSION 5.1 - READS DISCRETE TARGET FILE
C  MODIFIED FOR VERSION 5.1 TO USE DATA COORDINATES
```

```

C
C INPUT: TARGET FILE
C
C OUTPUT: POSITION, VELOCITY AND REFLECTIVITY OF TARGETS
C
C *****
C *****
SUBROUTINE RDDISC2
  USE DISC
  USE CLUT
  IMPLICIT INTEGER (I-N)
  READ(17,*)NSEG
  READ(17,*)(NND(I),I=1,NSEG)
  READ(17,*)((XXD(I,J),J=1,NND(I)),I=1,NSEG)
  READ(17,*)((YYD(I,J),J=1,NND(I)),I=1,NSEG)
  READ(17,*)((UPD(I,J),J=1,NND(I)),I=1,NSEG)
  READ(17,*)((RCSD(I,J),J=1,NND(I)),I=1,NSEG)
  READ(17,*)((VXD(I,J),J=1,NND(I)),I=1,NSEG)
  READ(17,*)((VYD(I,J),J=1,NND(I)),I=1,NSEG)
  WRITE(*,*)'DISCRETE TARGET FILE READ'
  RETURN
END

```

A sample discrete target file is furnished with the ADWRS 5.1 distribution disk.

3.0 PROGRAM STRUCTURE

A logical flow diagram of the main computer program is shown in Figure 3. At the beginning of the simulation, the input files are read into the program, and the variables to be used in the calculation are initialized.

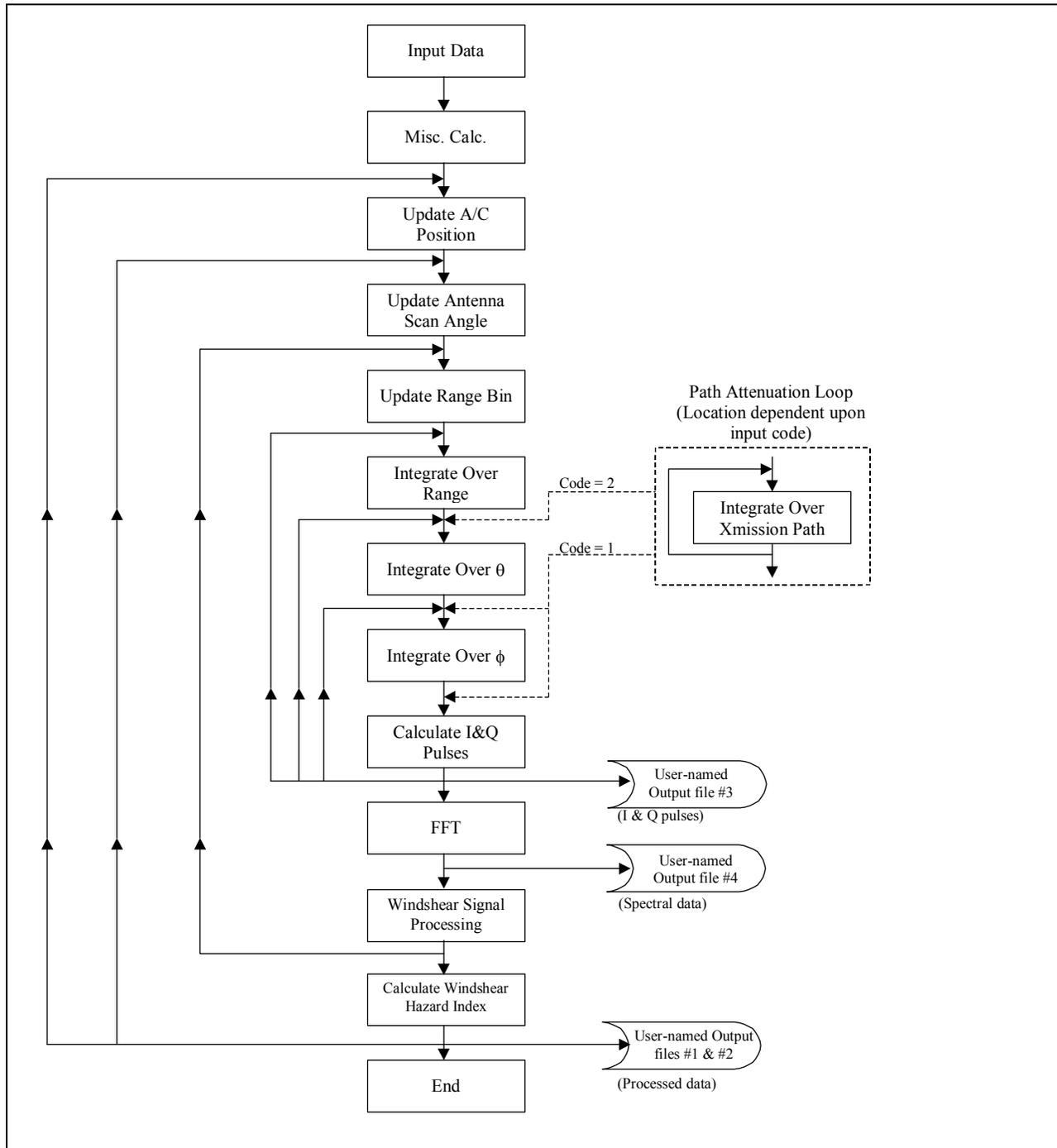


Figure 3. Logical Flow Diagram Showing the Main Calculation Loops in ADWRS.

The first loop in the program is the aircraft position loop, followed by the antenna azimuth scan angle loop as shown in Figure 3. Next is the range bin loop, which calculates the radar outputs in each range bin. The range bin loop is further subdivided into three integration loops for the integration over range, azimuth angle, and elevation angle of a segment of a circular hemisphere. The circular hemisphere is subdivided into small incremental volumes or areas representing the scatterers in the field of view. This breakdown of the range bin into incremental scatterers is sketched in Figure 4. The number of incremental cells in each range bin is specified in the input data file for range, azimuth and elevation.

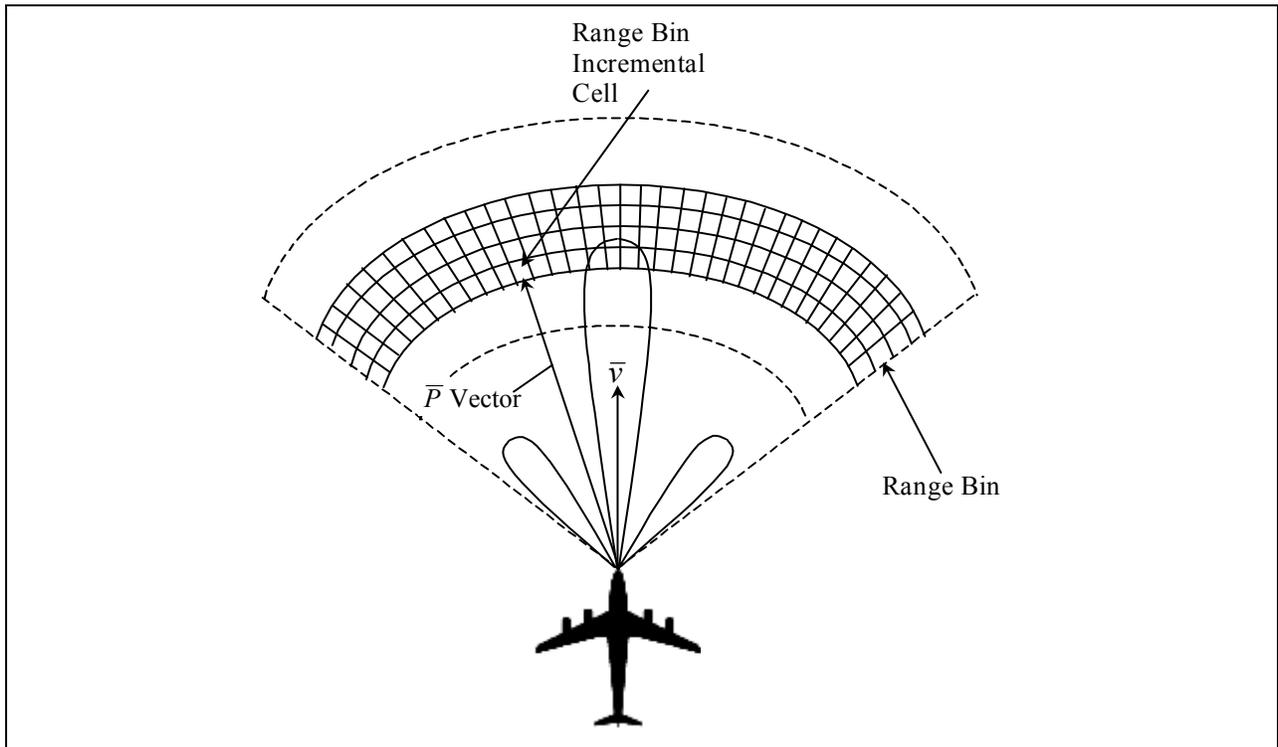


Figure 4. Breakdown of a Range Bin into Incremental Scattering Cells.

For each incremental cell in the range bin, the in-phase (I) and quadrature (Q) pulses are calculated using the equation given in Figure 5. In this equation, the amplitude of the return from the i^{th} scatterer is calculated (using the radar equation) with the scattering cross section obtained from either the ground clutter map or the weather model, as appropriate. Each scatterer in the range bin is assigned a random phase, a phase term representing the relative motion between the scatterer and the aircraft, and another normally distributed random phase term representing the system pulse-to-pulse phase error. This latter phase term is updated for each pulse, whereas the random phase associated with the scatterer is held constant over the series of

pulses used to derive the velocity. The sum of the returns from all of the incremental scatterers in the range bin provide a single I and Q pulse amplitude. Finally, a random variable representing receiver noise is added to the signal in accordance with the (input) specified noise level of the system. This procedure continues until a series of pulses (as specified in input data) are developed for further processing to derive the average velocity of the scatterers of the radar field of view.

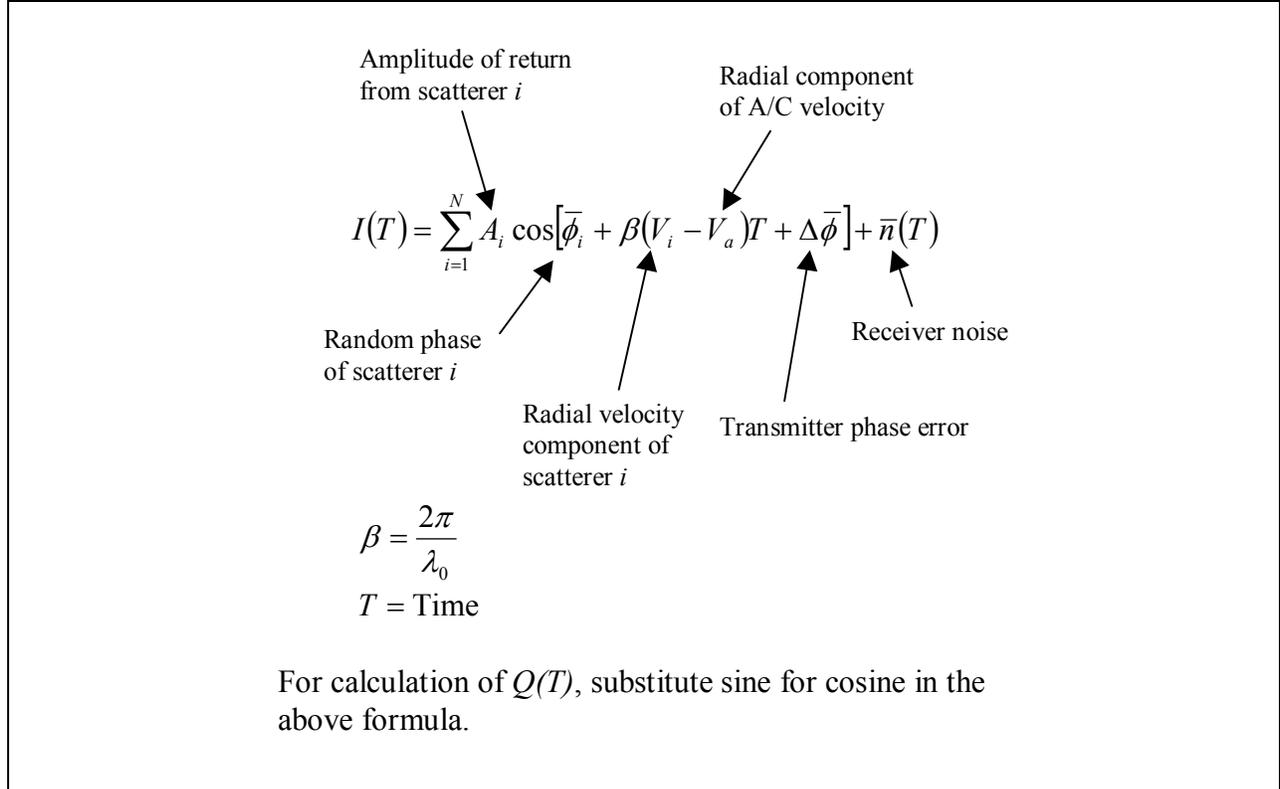


Figure 5. Techniques for Calculation of In-phase (I) and Quadrature (Q) Signals.

At this point, the series of I and Q pulses calculated by the simulation are output to an I and Q data file. These data may then be used for testing various signal processing algorithms for determining Doppler velocity, spectral width, or hazard index. The simulation continues by taking an FFT of the I and Q pulse stream to determine the Doppler spectra for the range bin. This Doppler spectra is also written to an output file, if desired.

To produce data products, various calculations are made on the I and Q pulse stream to determine power levels, average velocity using pulse-pair processing, spectral width using pulse-pair processing and spectral averaging, and other parameters. After this processing is complete, a new range bin is calculated as shown in the flow chart.

Upon completion of all range bins to be calculated, the wind shear hazard factor is calculated and added to the output file. In addition, data are output that permit development of simulated radar displays showing the radar outputs, (velocity, spectral width, or reflectivity) over a full azimuth scan of the radar.

3.1 Main Program

The main program in ADWRS is named ADWRS51.F. This program implements the functions described in the overview given above, with specialized functions implemented in the subroutines. The major function of ADWRS51.F is to implement the structure shown in Figure 3. Appendix A provides a commented listing of ADWRS51.F.

3.2 Subroutines

Table II lists the functions of the major subroutines used in the program. Appendix B provides detailed listings of these subroutines.

Table II. Functions of Major Subroutines in ADWRS

ADCONV	Calculates the AGC gain required to set the I and Q signals to a given average level and then quantizes the signals in accordance with the number of levels specified for the analog to digital converters. This subroutine assumes that AGC is implemented in each range bin separately.
ANTFLAT	Calculates the gain of a flat (Collins type) antenna with a beam width of 3.5 degrees. The algorithm implemented is obtained from data provided by Collins. Note that this algorithm is only valid for a frequency of 9.3 GHz.
ANTPAT	Calculates the antenna gain at a given input angle for a generic parabolic antenna of diameter specified in input data.
ANTREAL	Calculates the antenna gain using input angles and data from the file read in with READANT.
ATTEN	Calculates the two-way path loss due to moisture using moisture values obtained from the weather database.
AVRHAZ	Makes use of the OV array to average the hazard factor over several (as specified as input data) range bins.
BESSN	Finds the Bessel function of integer or fractional order for any input value greater than zero.
CALC_ANTENNA_GAIN	Calculates gain of radar antenna
CALC_ROTATION_MATRICES	Create matrices for converting from one coordinate system to another.
CLUT7	Calculates the value of sigma zero from the analytical ground

	clutter model.
CALC_CLUTTER_RETURNCLUTTER_SIGNAL_AMPLITUDE	Calculates rate of phase change of the ground clutter cell relative to the antenna due to motion i.e.- swaying of trees, current of river
CLUTTER_SIGNAL_AMPLITUDE	Calculates the amplitude of signal from ground clutter
DFILTER	Implements a high pass time-domain digital filter that operates on the I and Q signals and is used to suppress clutter signals.
DISCRETE_SIGNAL_AMPLITUDE	Calculates Amplitude Of Discrete Target Return
EFILTER	This subroutine is used as an alternate to the above and implements either a first or second order Butterworth filter to operate on the I and Q signals for suppression of ground clutter.
CALC_RAIN_RETURN ELEVATION_INCREMENTS	Calculate returns from weather
EMAT	Creates a multiple rotation operator matrix with rotation through the Euler angles (roll, pitch and yaw).
FFAC	Calculates the hazard index by differencing adjacent velocity values calculated along a range line. The hazard index is calculated from velocities derived by pulse pair processing, spectral averaging and true velocity.
FFRDR2	This subroutine implements a function used in the FFT subroutine to permute a complex data vector in reverse binary order to normal order.
FFT	Computes the fast Fourier transform of a complex vector of specified length.
FILCO	Calculates the coefficients necessary to implement the digital Butterworth filters that are calculated in subroutine EFILTER.
FILL_IQ_BINHEADER	Sets Values In Binheader Structure For Output File
FILL_IQ_FILEHEADER	Sets Values In Fileheader Structure For Output File
FILL_IQ_LINEHEADER	Sets Values In Lineheader Structure For Output File
FILTER_IQ_SIGNALS	Select and execute appropriate clutter filter.
FIND_THETA	Calculate Theta at center of integration cell
GAMMA	This subroutine is used to find the Gamma function for any variable greater than zero. Used in subroutine BESSN.
GET_WEATHER	Calculates the three Cartesian wind components of the weather given the spatial coordinates of a point. It also provides a reflectivity value and the standard deviation of velocity at the spatial location specified.
GETBW	Calculates the 3dB beam width of the selected antenna in degrees.
GMPRD	Multiplies N x M matrix by an M x L matrix to provide an N x L matrix.
GMTRA	Transposes an N x M matrix A to give a M x N matrix A^T .

HORIZ	Calculates the antenna pointing angle with respect to the horizon in degrees.
NORMAL	This routine provides a random number in accordance with a normal distribution with a standard deviation of unity.
PPP	Calculates average Doppler velocity and spectral width using pulse pair processing algorithms.
RAIN_SIGNAL_AMPLITUDE	Amplitude of signal from rain scatterer
READANT	Reads actual antenna pattern data on the antenna data file WXANT.dat.
RECEIVER_NOISE	Calculate power in returned radar signal added by receiver noise
ROT	Creates rotation operator matrix.
SETVAR	Places the variables to be output into an array called OV. The array OV is then written to an output file specified as input data.
SIGNAL_PROCESSING	Calculate average Doppler velocity and spectral width using pulse pair processing algorithms
SPHER	Converts a vector defined in xyz rectangular coordinates to a vector defined in standard spherical coordinates.
STRNUM	Generates a uniformly distributed random number between 0 and 1.
VELTRU	Calculates the "true" velocity component along the projected antenna beam center line. This true velocity component is then used in comparisons with radar calculated velocities.
WEATHER_PHASE_TERM	Calculates the rate of phase change of weather (wind, rain) due to relative motion along L.O.S.
WRITE_IQ_DATA	Sets Iq Values In Iq Structure For Output File And Writes Them To File
WXANAL	Calculates the average Doppler velocity in a range bin using a spectral averaging technique.

4.0 INPUT FILES

There are several input data files necessary to run the program. These files are discussed in the following sections (typical filenames are shown):

1. Parameter Definition file (WXIN40.DAT)
2. Filename definition file —3D Weather (WXFILES.DAT)
3. Weather data base (4 files, user-named)
4. XXX Discrete Target data file
5. XXX Antenna pattern data (WXANT.DAT)

4.1 Parameter Definition File

The parameter definition file provides the parameters that set up the simulation scenario. It consists of simulation parameters, radar parameters, parameters to define the weather and clutter characteristics, and parameters to define the signal processing algorithms to be used. Table III adds explanatory notes for the parameters in the file. Most of the inputs are self-explanatory, and comments are added where required.

Table III. Description of Parameter Definition Input File

AIRCRAFT POSITION & VEL	0.	
	0.	
A/C X-Offset from WX Origin (km)	3.	Aircraft X position with respect to origin of weather data.
A/C Y-Offset from WX Origin (km)	-.5	Aircraft Y position with respect to origin of weather data.
A/C Altitude (1000 ft)	13.	Aircraft altitude in units of 1000 ft.
A/C X-Velocity (kts)	0.	Aircraft X velocity component in knots.
A/C Y-Velocity (kts)	150.	Aircraft Y velocity component in knots.
A/C Rate-of-climb (ft/sec)	0.	
Roll Attitude (deg)	0.	Based on right-handed, u-v-w coordinate system. The u axis is along the aircraft velocity vector, and w is positive down.
Pitch Attitude (deg)	0.	
Yaw Attitude (deg)	0.	
A/C Weight (1000 lb)	177.2	Weight of aircraft at takeoff
	0.	
SIMULATION PARAMETERS	0.	
	0.	
Az Integration Range/2 (deg)	2.0	These five parameters define the integration range of the program with respect to the antenna beam center. Wider ranges provide more accuracy but require longer running times. The angles should be large enough to take into account the major sidelobes of the antenna for best accuracy. The angular increments should be much less than the antenna beam-width (approximately 3 degrees). The number of radar scatterers used in the calculation are defined by the volume specified by the pulse width and angular ranges and the associated increments defined above.
Az Integration Increment (deg)	.3	
El Integration Range/2 (deg)	1.0	
El Integration Increment (deg)	.2	
Rng Integration Increment (m)	100.	
Random Number Seed (0-1)	.224	Value used to initial random number generators.
Subtract Aircraft Vel. (1=Y)	1.	Subtract aircraft velocity so that derived Doppler frequencies are referenced to a value of zero.
No. of Complete Raster Scans	2.	Desired number of azimuth scans
Seconds Between Bar Sequences	5.	Time interval between scans
Diagnostic Flag (1=ON)	0.	
Spare	0	
	0.	

WEATHER & CLUTTER	0.	
	0.	
Rain Standard Deviation (m/s)	1.	Simulates fine-grained turbulence. Used with clutter S. D. to define random velocity component that is applied to the calculated radial component of the velocity of a scatterer. Random component is different for each radar pulse.
Clutter Standard Deviation (m/s)	0.5	Simulates moving grass, trees, etc. Used with rain S. D. to define random velocity component that is applied to the calculated radial component of the velocity of a scatterer. Random component is different for each radar pulse.
Clutter Calc. Flag (1=ON, 0=OFF)	1.	“Switch” to turn off calculation of ground clutter returns.
Discrete Calc. Flag (1=ON, 0=OFF)	0.	“Switch” to turn off calculation of discrete target returns.
Reflectivity Calc. Flag (1=ON, 0=OFF)	1.	“Switch” to turn off calculation of reflectivity.
Reflectivity Calc. Thres. (dBz)	-14.	The radar return from the rain scatterer is not calculated if the reflectivity of the scatterer (from the weather database) is below this threshold.
Minimum Reflectivity (dBz)	-20.	Defines the reflectivity "floor" for the weather database. For example, if the database provides a reflectivity of -40 dBz, the value will be reset to the number provided (-25 dBz).
Ground Clutter Type Code	5.	Specifies type of clutter for analytical clutter model: (1=soil/sand, 2=grass, 3=tall grass/crops, 4=trees, 5=urban, 6=wet snow, 7=dry snow)
Surface Std. Dev. (m)	.5	Standard deviation of ground surface (for clutter model).
Use TKE data (1=Y)	0.	“Switch” to turn off use of TKE database.
Add dBZ to Data (dBZ)	0.	Adds a constant dBZ value to reflectivity database (used to elevate reflectivity to values that can be seen by current weather radar).
	0.	
RADAR PARAMETERS	0.	
	0.	
Initial Radar Range (km)	1.	
Number of Range Bins	50.	

Transmitted Power (watts)	150.	
Frequency (GHz)	9.3	When using antenna #1, frequencies can be specified between 5-20 GHz.
Pulse Width (microsecs)	1.5	Radar Pulse Width
Pulse Repetition Interval (microsecs)	268.6	Interval between radar pulses
Receiver Noise Figure (dB)	3.	Estimated receiver noise figure
Receiver Losses (dB)	2.	Estimated losses- antenna to receiver
RMS Trans. Phase Jitter (deg)	.2	Specify transmitter/receiver phase stability.
RMS Trans. Freq. Jitter (Hz)	0.	Specify transmitter/receiver frequency stability.
Pulse Sampling Interval (us)	1.0	Interval between samples of the returned pulse. This determines bin size.
	0.	
ANTENNA & RASTER SCAN	0.	
	0.	
Antenna Type	3.	See code below
Type 1		Generic parabolic antenna
Type 2		Rockwell-Collins type flat antenna. Can only be used at 9.3 GHz.
Type 3		Uses an input data set consisting of measured pattern data of a Rockwell-Collins flat antenna. Can only be used at 9.3 GHz.
Antenna Radius (m)	.381	For parabolic antenna (type 1) only.
Aperture Taper Parameter	.316	For parabolic antenna (type 1) only.
Scan Type (1=Az/El, 2=El/Az)	1.	Currently, only option 1 (Az/El) is implemented.
Scan Rate (deg/sec)	20.	Not currently used.
Az Scan Center (deg)	0.	Center of azimuth scan.
El Scan Center (deg)	0.	Center of elevation scan.
No. of Az (El) Scan Lines (odd)	46.	Number of azimuth lines.
No. of El (Az) Bars	2.	Number of different elevations used.
Az Scan Increment (deg)	1.	Angular increment between scan lines.
El Scan Increment (deg)	-2.	Angular increment between elevation bars.
Spare	0.	
	0.	
SIGNAL PROCESSING	0.	
	0.	
Number of Pulses/Scan Line	128.	Number of pulses averaged to derive velocity (pulse-pair processing) and number of pulses used to determine Doppler velocity spectra (for spectral averaging derivation of velocity).
Number of A/D bits	12.	Number of bits used in the simulated A/D converter.

AGC Gain Factor	.6	The level to set the AGC gain for a range bin. For example, if the average signal level in a range bin is S_a , the product of the AGC gain and S_a is made to be equal to the gain factor specified. The simulated A/D sampling range of the resulting signal is -1.0 to 1.0.
Processing Threshold (dB)	4.	Sets threshold relative to receiver noise above which velocity and hazard factor calculations are made. If the signal (power level above noise) is below this value, velocity and hazard factor are set to 0.
Clutter Filter Code (-2 to N)	2.	Defines type of time-domain clutter filter. Not applicable if clutter turned off.
The clutter code is:		
-2		Two-pole Butterworth filter
-1		Single-pole Butterworth filter
0		No Filter
1		Single pole high-pass filter
2		Two single-pole filters cascaded
.		Three single-pole filters cascaded
.		Etc
Clutter Filter Vel. Cutoff (m/s)	3.	Defines the filter cutoff in velocity units. Not applicable if clutter turned off.
No. of Bins for F-factor Avr.	5.	Number of range bins used to average the hazard factor. Values are 0, 3, and 5 depending on desired smoothing.

Threshold for F-factor Algorithm	1.0	Threshold for the weighted least-squares hazard factor calculation algorithm. The algorithm fits a least squares line to a series of 5 velocity measurements and determines the slope of the line. If the average weighted residual (velocity measurement deviation/spectral width) is greater than the specified threshold, the hazard measurement is considered invalid, and the F-factor is set to zero.																		
Spare	0.																			
Spare	0.																			
	0.																			
DATA PRODUCTS	0.																			
	0.																			
Alarm Program Data Code	1.	Define the amount and type of data products (output files). The code for each data product defines the scan and lines for which data will be written for each file. Data will be written for: <table border="0"> <thead> <tr> <th><u>Code</u></th> <th><u>Scan No.</u></th> <th><u>Line No.</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>None</td> <td>None</td> </tr> <tr> <td>1</td> <td>All</td> <td>All</td> </tr> <tr> <td>2</td> <td>All</td> <td>Start to Finish + 1</td> </tr> <tr> <td>3</td> <td>Start to Finish + 1</td> <td>All</td> </tr> <tr> <td>4</td> <td>Start to Finish + 1</td> <td>Start to Finish + 1</td> </tr> </tbody> </table>	<u>Code</u>	<u>Scan No.</u>	<u>Line No.</u>	0	None	None	1	All	All	2	All	Start to Finish + 1	3	Start to Finish + 1	All	4	Start to Finish + 1	Start to Finish + 1
<u>Code</u>	<u>Scan No.</u>		<u>Line No.</u>																	
0	None		None																	
1	All		All																	
2	All		Start to Finish + 1																	
3	Start to Finish + 1		All																	
4	Start to Finish + 1		Start to Finish + 1																	
C.S.V. (Plot) Output Code	4.																			
IQ Data Output Code	4.																			
Spare	0.																			
Spare	0.																			
Fourier Spectra Data Code	4.																			
Start Scan Number	1.																			
Finish Scan Number	2.																			
Start Line Number	7.																			
Finish Line Number	8.																			
Spare	0.																			

4.2 Filename Definition File

This file provides to the simulation a list of the filenames for the 3D weather. The structure of the file is shown in Figure 6.

Line 1	3D weather database containing u-component of wind velocity.
Line 2	3D weather database containing v-component of wind velocity.
Line 3	3D weather database containing w-component of wind velocity.
Line 4	3D weather database containing radar reflectivity.
Line 5	3D weather database containing subgrid standard deviation of velocity.
Line 6	Discrete target filename.
Line 7	Unformatted user-named post-processing output file containing data for the Post-Processing, Analysis & Display program.
Line 8	Formatted user-named processed data output file containing (powers, velocities, etc.).
Line 9	Formatted user-named I & Q data output file.
Line 10	Formatted user-named FFT data output file.

Figure 6. Filename Definition File

4.3 Weather Data Input Files (3D)

Three-dimensional weather simulations require four data input files for turbulence. The naming convention of these files is left to the user; however, the data should be furnished as formatted ASCII files, with one file each for:

- Wind velocity in the U direction in m/s
- Wind velocity in the V direction in m/s
- Wind velocity in the W direction in m/s
- Radar reflectivity in dBZ

Data files must each include three header lines at the beginning of the file which contain information in a format similar to the following:

```
NX= 256  NY= 256  NZ= 128 LO= 500 SIGMA= 3
DX= 100.0  DY= 100.0  DZ = 100.0
XSTART = 0.0000E+03 YSTART = 2.0000E+03 ZSTART = 5.0000E+03
```

NX, NY, and NZ specify the number of grid points in the u, v, and w, directions respectively. Values for LO and sigma are for information purposes only and are not used. If this information is not available, a dummy value should be entered. DX, DY, and DZ specify the distance between grid points. Zstart is the starting elevation for data on the weather grid. XSTART and YSTART give the starting position of the simulated aircraft. The header for each of the five data files used should contain identical information.

Following the header information are the actual data values that have been multiplied by a value of 10.0 with any trailing decimals truncated. This was done to allow the use of smaller variables without significant loss of information. The data in each of the files must be relative to a common origin. Each file will contain information for a matrix with dimensions NX x NY x NZ. Data must be written in the file in such a manner that the following Fortran statement may read it:

```
read(9,*) ((( U(I,J,K), I=1,NX),J=1,NY),K=1,NZ)
```

A program is provided with this release of ADWRS to facilitate the conversion of data from TASS format to the appropriate format. The FORTRAN program Tassreader.f90 is a simple data processor which is used to convert data provided by TASS into a format appropriate for input into ADWRS. The program accepts one input file containing the names of eight data files. The first four names designate existing input files which contain weather data. Input files must be in the same format as the TASS 191-06 phase three dataset which uses a 25m grid. A separate TKE file is not required. The next four names designate output files to be created by the program and filled with reformatted output data. The program produces four separate output files, each with an appropriate header. These output files can in turn be used as input to ADWRS 5.1 by placing their names in the filename configuration file.

4.4 Antenna Pattern Data File

This input file consists of measured antenna pattern data taken in the NASA/LARC anechoic chamber. The data consists of pattern data taken in the two principle planes of the

antenna at 0.1-degree increments over a range of 180 degrees. This actual pattern data is used in the simulation if antenna option #3 is chosen in the parameter definition input. Note that when this antenna is used in the simulation, the results are only valid for a frequency of 9.3 GHz.

5.0 GRAPHICAL USER INTERFACE (GUI)

ADWRS can be operated by three different methods. It may be executed from the command prompt using the following syntax:

```
ADWRS parameter_definition_file filename_definition_file
```

For example, using typical filenames: ADWRS WXIN40.DAT WXFILES.DAT

ADWRS can also be executed from within the Visual Fortran Development Environment by accessing the project menu, selecting the settings option, and then the Fortran tab. The category box should have "General" selected. In the text box labelled "Program arguments", users should enter the command line syntax shown above for the command line option including pathnames if required.

A third method of execution is to use the Graphical User Interface (GUI) program. This program was developed in Visual Basic to assist users in creating and changing input files with a minimum of errors. The GUI provides interactive help, direct access to this User's Guide, and default values for variables.

A screen shot of the ADWRS GUI main menu is shown in figure 7. This is the first interface a user sees when activating the GUI. Figures 8, 9 and 10 are screenshots of the submenus available. Explanations and definitions for various parameters are available by moving the cursor over the parameter name. This will cause context-sensitive help to appear on the screen. More detailed explanations are available by clicking on Help at the top of the main menu. This will activate an embedded copy of this User's Guide. Acrobat 4.0 must be present on the user's computer to activate the embedded document.

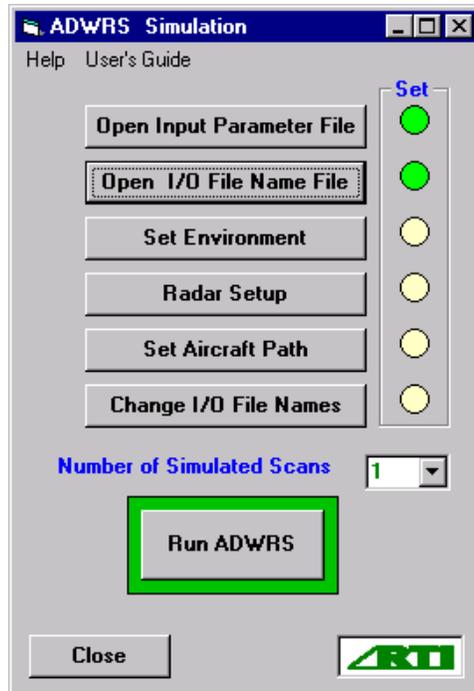


Figure 7. ADWRS GUI Main Menu

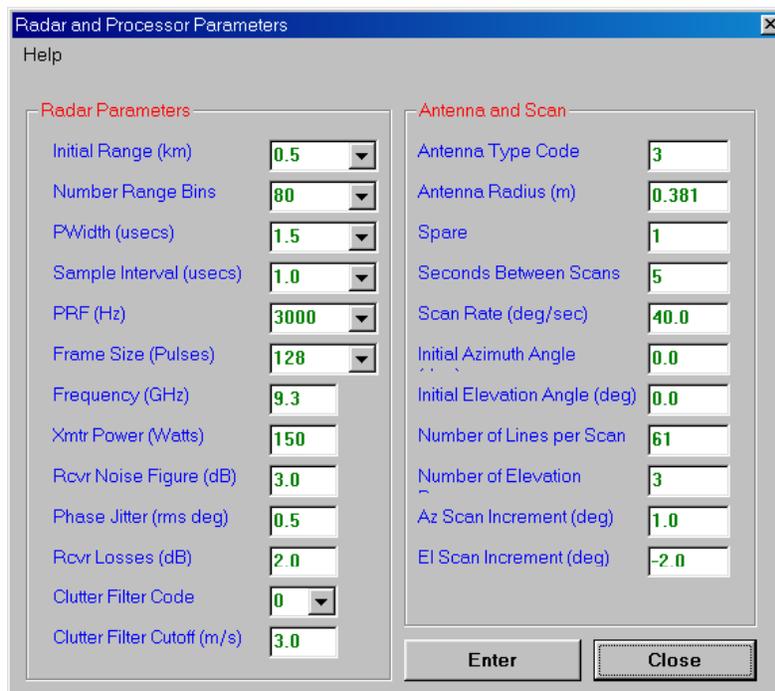


Figure 8. Radar Parameter Submenu

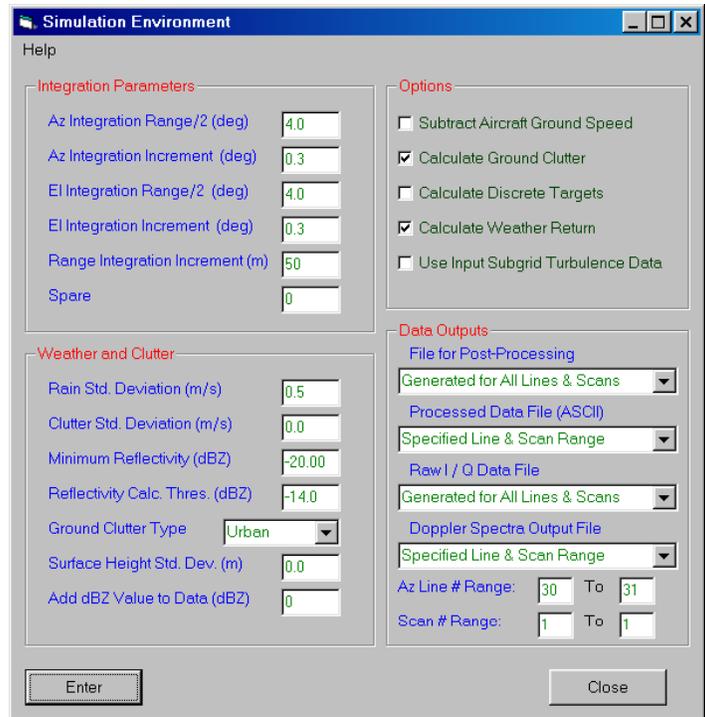


Figure 9. Simulation Parameter Submenu

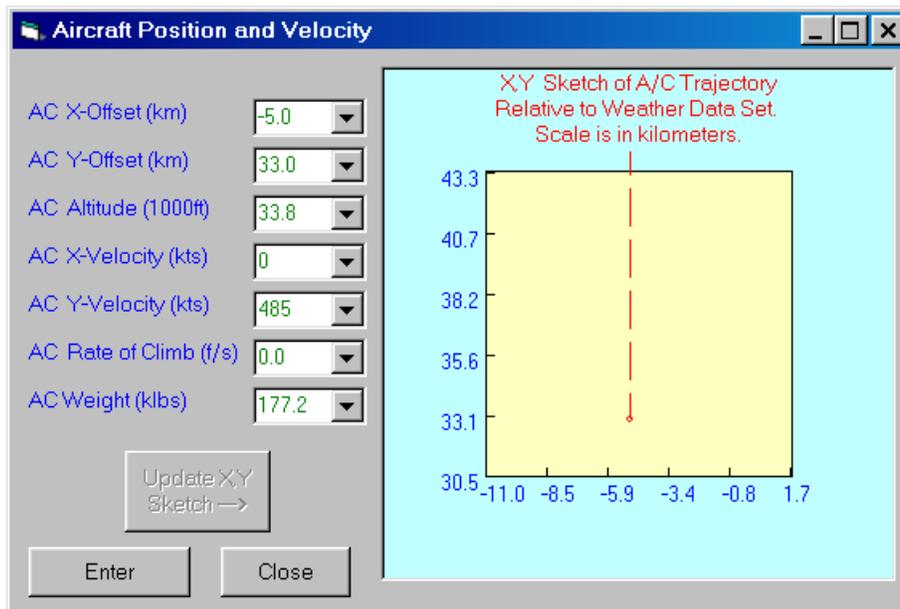


Figure 10. Aircraft Parameter Submenu

6.0 OUTPUT FILES

ADWRS automatically creates four user-named output files. The filenames are specified in the filename definition input file. (See Table III.)

6.1 Post Processing Data Output File

This file contains binary IQ data along with aircraft and radar parameters specifically needed by the Post Processing and Analysis program.. The file contains fileheaders, line headers, bin headers, and data. There is only one fileheader in the file. It contains the information described in Table IV.

Table IV. File Header for Post Processing Data Output File

<u>Variable name</u>	<u>Units</u>	<u>Description</u>
CDATE		MM/DD/YY
START_TIME_HR		Always zero
START_TIME_MIN		Always zero
START_TIME_SEC		Always zero
SPARE		
CLABEL		"ADWR"
NUM_SCANS		Number of scans simulated
NUM_LINES		Number of lines/scan simulated
NUM_SCANS_OUTPUT		Number of scans written to ouput file
NUM_LINES_OUTPUT		Number of lines/scan written to output file
NUM_RANGE_BINS		
NUM_PULSES		
PULSE_REP_FREQ	pulses/sec	
PULSE_WIDTH	meters	
SCAN_CENTER	deg	
SCAN_RANGE	deg	
SCAN_INCREMENT	deg	
SWEEP_SPEED	deg/sec	

TILT_CENTER	deg	Relative to horizon
TILT_RANGE	deg	
TILT_INCREMENT	deg	
TOTAL_BARS		
RADAR_GATE_DELAY	meters	Time from transmission of pulse until first sample recieved
BIN_WIDTH	meters	
AC_GND_SPEED	m/s	
RADAR_FREQ	Hz	
AC_WEIGHT	pounds	

The fileheader is followed by one or more line headers depending on the number of output lines specified by the user in the configuration file. A line header is written for each line specified and contains the following information:

Table V. Line Header for Post Processing Data Output File

<u>Variable name</u>	<u>Units</u>	<u>Description</u>
LINELABEL	char	"LINE"
TIME_HR	hours	
TIME_MIN	minutes	
TIME_SEC	seconds	
SPARE		
SCAN_INDEX		
LINE_INDEX		
STARTING_RANGE	meters	
SCAN_ANGLE	deg	
TILT_ANGLE	deg	
CURRENT_BAR		Bar count ID
AC_XCOOR	meters	
AC_YCORR	meters	

AC_ZCOOR	meters	
AC_LATITUDE	deg	Unused
AC_LONGITUDE	deg	Unused
AC_HEADING	deg	
AC_TRACK	deg	Same as heading
AC_AIRSPEED	meters/sec	Ground speed modulated by weather data

Each line header is followed by a series of range bin headers and data. There is one range bin header for each range bin specified by the user in the configuration file, and each range bin header is followed by the IQ data calculated for that bin. An I and Q value are calculated for each radar pulse.

Table VI. Bin header for Post Processing Data Output File

<u>Variable name</u>	<u>Units</u>	<u>Description</u>
BIN_NUMBER		
RANGE_TO_BIN	meters	

Table VII. Data for Post Processing Data Output File

<u>Variable name</u>	<u>Units</u>	<u>Description</u>
IDAT	VOLTS	In-Phase value of data
QDAT	VOLTS	Quadrature value of data

6.2 Processed Data Output File — Comma Separated Values

This output file contains the main output parameters of the simulation and consists of twenty-four variables as shown in Table VIII. The file is formatted (ASCII) as comma separated values (CSV) for use in plotting programs or spreadsheets.

Table VIII. Contents of Processed Data Output File

Variable Number	Variable	Units
(1)	Radar Range	km
(2)	Velocity from Pulse-Pair Processing	m/s
(3)	Velocity from Spectral Average	m/s
(4)	True Velocity along Antenna Boresight	m/s
(5)	True Vertical Wind Velocity	m/s
(6)	Power Return from Noise	dBw
(7)	Power Return from Rain	dBw
(8)	Power Return from Clutter before Filter	dBw
(9)	Power Return from Clutter after Filter	dBw
(10)	Power from Discretes	dBw
(11)	Total Power Received	dBw
(12)	AGC Gain (Automatic Gain Control)	dB
(13)	Reflectivity	dBz
(14)	Spectral Width from Pulse-Pair Processing	m/s
(15)	Hazard Factor from Pulse-Pair Processing (Horizontal Component)	
(16)	Hazard Factor from Spectral Average (Horizontal Component)	
(17)	Doppler Velocity from Pulse Pair Processing, Unfiltered	m/s
(18)	True Hazard Factor (Horizontal Component)	
(19)	Hazard Factor (Vertical Component)	
(20)	True Hazard Factor (Horizontal + Vertical Comp.)	
(21)	Spectral Width, Unfiltered	
(22)	Spare	
(23)	Spare	
(24)	Spare	

These data are output in binary format and are preceded by headers as follows:

Table IX. Headers for Processed Data Output File

File Header:	(Written once each file)
	<u>Description</u>
	No. of scans written to this file
	No. of lines per scan written
	Scans in data run
	Lines in each scan
	No. of range bins

	No. of pulses
	Aircraft velocity (m/s)
	Pulse repetition rate (sec)
	Azimuth scan range (deg)
	Azimuth scan increment (deg)
	Range bin size (m)
	Time between scans (sec)
Line Header:	(Written once each range line)
	<u>Description</u>
	Scan Index
	Line Index
	Range to integration cell
	Antenna azimuth (deg)
	Antenna elevation (deg)

6.3 I and Q Data Output File

This file contains the radar I and Q pulse values for each range bin along a range line and is output as formatted (ASCII) text. The file contains file headers and data as follows:

Table X. Headers and Data Contents for I & Q Data Output File

File Header:	(written once each file)
	<u>Description</u>
	No. of scans written to this file
	No. of lines per scan written
	Scans in data run
	Lines in each scan
	No. of range bins
	No. of pulses

	Aircraft velocity (m/s)
	Pulse repetition rate (sec)
	Azimuth scan range (deg)
	Azimuth scan increment (deg)
	Range bin size (m)
	Time between scans (sec)
Line Header:	(Written for each range bin)
	<u>Description</u>
	Scan Index
	Line Index
	Range Bin Index
	Range to integration cell (km)
	Radar range (km)
	Antenna azimuth (deg)
	Antenna elevation (deg)
	Coordinate of A/C position in x direction
	Coordinate of A/C position in y direction
	Coordinate of A/C position in z direction
Data:	(For each pulse simulated)
	<u>Description</u>
	I value—unfiltered
	Q value—unfiltered
	I value—filtered
	Q value—filtered

6.4 Doppler Velocity Spectra Output File

This file contains two file headers similar to the I & Q data file. The data written for each spectral line in each range bin consist of Doppler Spectra derived from the I & Q signals using a Fourier Transform as follows:

Table XI. Contents for Spectral Data Output File

File Header:	(written once each file)
	<u>Description</u>
	No. of scans written to this file
	No. of lines per scan written
	Scans in data run
	Lines in each scan
	No. of range bins
	No. of pulses
	Aircraft velocity (m/s)
	Pulse repetition rate (sec)
	Azimuth scan range (deg)
	Azimuth scan increment (deg)
	Range bin size (m)
	Time between scans (sec)
Line Header:	(Written for each range bin)
	<u>Description</u>
	Scan Index
	Line Index
	Range Bin Index
	Range to integration cell (km)
	Radar range (km)
	Antenna azimuth (deg)
	Antenna elevation (deg)
	Coordinate of A/C position in x direction

	Coordinate of A/C position in y direction
	Coordinate of A/C position in z direction
Data:	(For each spectral line)
	<u>Description</u>
	Velocity (m/s)
	Filtered spectral line magnitude
	Unfiltered spectral line magnitude

The number of spectral lines written for each range bin is equal to the number of pulses specified in the Parameter Definition input file. The file is output as formatted (ASCII).

REFERENCES

- [1] Doviak, R. J.; and Zrnic, D. S.: "Doppler Radar and Weather Observations." Academic Press Inc., 1984.
- [2] Proctor, F. H.: "The Terminal Area Simulation System Vol. I: Theoretical Formulation." NASA CR 4046: DOT/FAA/FM-86/50, I. April 1987.
- [3] Proctor, F. H.: "The Terminal Area Simulation System Vol. II: Verification Cases." NASA CR 4047: DOT/FAA/PM-86/50, II. April 1987.
- [4] Eaves, J. L.; and Reedy, E. K.: "Principles of Modern Radar." Van Nostrand Reinhold, 1987.
- [5] Ruck, G.; Barrick, D.; Stuart, W.; Krichbaum, C.: "Radar Cross Section Handbook Vol. II." New York: Plenum, 1970.
- [6] Britt, C. L.; Kelly, C. W.; and Wissel, V. L.: "Verification of IQ Signal Characteristics Generated by ADWRS V4.0" RTI Report 7473/014-02S, January, 2000.

APPENDIX A

MAIN PROGRAM LISTING

PROGRAM ADWRS51

```
C*****
C
C      TURBULENCE RADAR SIMULATION PROGRAM (VERSION 5.1)
C
C      THIS PROGRAM WAS WRITTEN FOR NASA, LANGLEY RESEARCH
C      CENTER UNDER CONTRACT NAS1-17639 BY:
C
C          CHARLES L. BRITT, Ph.D
C          RESEARCH TRIANGLE INSTITUTE
C          1 ENTERPRISE PKWY,SUITE 310
C          HAMPTON, VA 23666
C          Ph. 757-827-1160
C
C      REVISION 8/9/99      EDITING CHANGES - VERSION 3.1
C                          Added analytical clutter model
C                          REMOVE UNUSED CODE & FUNCTIONS
C      REVISION 11/2/99    ADDED ADDITIONAL INPUT PARAMETERS
C                          CHANGED METHOD OF READING INPUT FILE
C      REVISION 11/21/99   DEFINED INPUT PARAMETERS THAT WERE
C                          UNDEFINED IN WXIN9A. ALSO
C                          MADE VARIOUS ERROR CORRECTIONS AND ADDED
C                          DIAGNOSTIC SCREEN PRINTOUTS
C      REVISION 12/3-5/99  MANY REVISIONS TO GET THE PROGRAM RUNNING
C                          WITH THE NCAR DATA FILE & NEW WEATHER SUB
C                          ADDED PARAMETERS TO INPUT FILE AND RENAMED
C      REVISION 12/5-8/99  OUTPUT FILES CHANGED TO FORMATTED FILES. ADD
C                          UNFILTERED VELOCITY TO OF1.DAT AND OF2.DAT
C                          SEVERAL OTHER MISC CHANGES - REMOVED
C                          AUTOREGRESSIVE PROCESSING
C      REVISION 10/30/01   MAJOR RESTRUCTURING OF CODE.
C                          CAN USE TASS (PRE-PROCESSED) DATA.
C                          DATABASES CAN BE ARBITRARY DIMENSIONS.
C                          WEATHER DATA STORED AS INTEGER*2.
C                          ADDED XSTART, YSTART AND ZSTART VARIABLES.
C      REVISION 1/30/02   ADDED OUTPUT FILE #5 TO SUPPLY UNFORMATTED
C                          RADAR DATA TO DISPLAY. OTHER CHANGES TO
C                          GET RID OF OLD FORTRAN CODING TECHNIQUES AND
C                          RESTRUCTURE INTEGRATION LOOPS (VER 5.1)
C*****
C*****
```

```
USE WIND3D ! USED BY MAIN,READ_DATABASES,GET_WEATHER_PARAMETERS
USE DISC  ! USED BY MAIN,RDDISC
USE WIND  ! USED BY MAIN,BRTSWD2
USE CONST ! USED BY MAIN,WXANAL,PPP,ANTREAL,ANTPAT
USE ANT   ! USED BY MAIN,ANTPAT
USE ATTN  ! USED BY MAIN,ATTEN
C      USE BFIL ! USED BY FILCO AND EFILTER (NOT MAIN)
C      USE ANTR ! USED BY READANT AND ANTREAL
USE CLUT  ! USED BY MAIN.RDDISC
```

IMPLICIT INTEGER (I-N)

LOGICAL DBZ_STATUS

INTEGER SUBTRACT_AC_velocity,DO_CLUTTER_FLAG,DO_DISCRETES_FLAG,
2 DO_DBZ_FLAG,CLUTTER_TYPE,ANTENNA_TYPE,SCAN_TYPE

```

2    INTEGER PHIS_PER_SUM, THETAS_PER_SUM, RANGES_PER_SUM, NUM_AD_BITS,
      BIN_NUMBER, SCAN_INDEX
INTEGER NUM_LINES_IN_OUTPUT, NUM_SCANS_IN_OUTPUT, AZSCAN_INDEX
REAL    KW, ZO, BX, BY, PSI, GAIN, ANTANG, XLOSS, AZ, EL, BINSZ
CHARACTER(4) CTEST
CHARACTER(80) fn_file, cfile
CHARACTER(50) FNAME
CHARACTER(80) UFILE, WFILE, RFILE, DFILE, OF1, OF2, OF3
CHARACTER(80) OF4, VFILE, TKE_FILE
DIMENSION R1(3,3), R2(3,3), R3(3,3), R4(3,3), R5(3,3)
DIMENSION R6(3,3), R7(3,3), RT(3,3), RE(3,3), RFIN(3,3)
DIMENSION RHO(3), RHOS(3), OV(24,128)
DIMENSION RV(3,3), VU(3), VX(3), RHOA(3), RFIN2(3,3)
DIMENSION PCLUT(512), PRAIN(512), PTOT(512), PNSE(512), PDISC(512)
DIMENSION CC1(512), CC2(512)
DIMENSION EE(ICDIM), VVV(ICDIM), AAA(ICDIM), VP(3)
DIMENSION E(ISDIM), VV(ISDIM), AA(ISDIM), VSAVE(3,2), RP(3)
DIMENSION U1(512), U2(512), IMAGDB(512), IDATA(512), RPS(3)
DIMENSION AAD(IDDIM), EED(IDDIM), VVD(IDDIM)
DIMENSION U1R(512), U2R(512), IMAGDB2(512), IDATA2(512)

REAL IP_READ, IP_GET, WEIGHT, TIME_SECS, SCNRATE
REAL AXX,  AXXX,  AYY,  AYYY,  AZZ,  ZZZ
REAL STHT, CTHT, SUM1, SUM2, TT, XMIT_JITTER, SUM1C, SUM2C, CC1, CC2
REAL SUM1D, SUM2D, SUMZZ, INITIAL_RANGE, DEL_TIME, DEL_AZSCAN
REAL DEL_ELSCAN, SLANT_RANGE, DEL_PHI, VEL_STD_DEV, PWIDTHM, RGD
REAL SDNSE2, SDNSE
REAL AC_VELOCITY, AC_AIRSPEED, AC_HEADING
INTEGER KK, K, KKD, KDMAX, KTMAX, KMAX, KKMAX, KNTZ, II, KT, I, J
INTEGER IELKNT, NBARS, KLIM, KKLIM, KDLIM
INTEGER IALMCD, ICSVCD, IQCODE, IFTCODE

INCLUDE 'ADWRS51.FH'

TYPE(FILEHEAD) FHEAD
TYPE(LINEHEAD) LHEAD
TYPE(BINHEAD) BHEAD
C
C DATA AG, XNMTM, XKTTMS, XKMTM, BOLTZMAN_CONSTANT, XTO/
2    9.80616, 1853.2, 0.5148, 1000.0,
3    1.38E-23, 290.0/
C
C *****INPUT DATA*****
C
CALL GETARG(1, cfile)
CALL GETARG(2, fn_file)
OPEN(8, FILE=cfile, STATUS='OLD') ! open parameter file

WRITE(*,*) 'READING INPUT DATA'
XII=IP_READ(8) ! read configuration file

USE_TKE_FLAG = INT(IP_GET(10,3)) !flag to use TKE data

CALL READ_FILENAMES

CLOSE(8)

```

```

C      ****MISC  CONSTANTS****

      PI2=2.0*PI          !
      KW=sqrt(0.9313)    ! complex refractive index of water
                        ! (Doviac & Zrnic page 27)
      FT_2_METERS=0.3048 ! constant to convert feet to meters
      RANDOM_NUMBER=0.304 !
      TEST=DEGREES_2_RAD !FOR DEBUG
      KLIM=ISDIM         !LIMITS ON SCATTERER DIMENSIONS
      KKLIM=ICDIM
      KDLIM=IDDIM

C      **** READ CONFIGURATION DATA ****

      CALL READ_PARAMETER_FILE

      GLIDESLOPE=0.0      !GLIDESLOPE ANGLE
      ANGM = 0.0          !WINDFIELD ROTATION ANGLE
      EL=ELSCAN_CENTER   !ELEVATION ANGLE
      DUM=0.0

      IF (UNIFORM_RANDOM_NUM.EQ.0.) UNIFORM_RANDOM_NUM=.352 !1/9/02 CLB

      CALL OPEN_FILES

C      ****MISC CONVERSIONS****

      AZSCAN_RANGE=( (NLINE-1)/2)*DEL_AZSCAN

      IF (NLINE .EQ. 1) THEN
        ISLINE=1
        IFLINE=1
      ENDIF
      IF (NUM_SCANS.EQ.1) THEN
        ISSCAN=1
        IFSCAN=1
      ENDIF

      XNMTM=1853.2
      ALPHA=10.** (2.660*ALOG10(RADAR_FREQ_GHZ)-4.5631) !ATTEN CONSTANTS
      BETA=10.** (-.1669*ALOG10(RADAR_FREQ_GHZ)+.2586)
      CON2=ALPHA/4.34E3
      IATTN=2          ! when to apply attenuation to processing
      IF (XIFIL .GE. 0.0) IFIL=INT(XIFIL + .1) ! filter code
      IF (XIFIL .LT. 0.0) IFIL=-1*INT(ABS(XIFIL) + 0.1)
      PFAC=10.** (PTHRES/10.) ! PROCESSING THRESHOLD (LINEAR)
      ZFAC=10.** (ZTHRES/10.) ! REFLECTIVITY THRESHOLD (LINEAR)
      NLEV=INT(2.0**(NUM_AD_BITS-1.) + 0.1) !number of possible levels in
a/d

      THETA_MIN=-1.0*THETA_MAX
      THETAS_PER_SUM=INT((THETA_MAX-THETA_MIN)/DEL_THETA)
      AC_VELOCITY=SQRT(XVEL*XVEL + YVEL*YVEL)
      FREQ=RADAR_FREQ_GHZ*1.E9
      PWIDTH=PULSE_WIDTH_USEC*1.E-6
      PWIDTHM=PWIDTH*CC/2.0
      PULSE_INTERVAL=PULSE_INTERVAL_USEC*1.E-6
      PULSE_RATE=1./PULSE_INTERVAL

```

```

SAMPLE_INTERVAL=SAMPLE_INTERVAL_USEC*1.E-6
XLAM=CC/FREQ
C1=PI2*ANTENNA_RADIUS*ANTENNA_RADIUS
C2=2.*PI2/(XLAM*XLAM) ! constant to convert vel. to phase change rate
C12=C1*C2
XK=PI2/XLAM
GAINO=C12/2.
RCVR_LOSS=10.**(RCVR_LOSS_DB/10.)
CU=4.*PI*FREQ/CC ! CONSTANT FOR DOPPLER CALCULATION

LINE_COUNT=0
ISCAN_COUNT=1
SEC_PER_LINE=0.
IF(SCNRATE.NE.0.)SEC_PER_LINE=DEL_AZSCAN/SCNRATE
SEC_PER_SCAN=DEL_TIME !TIME BETWEEN SCANS
TIME_SECS=0.

PSIANGR=ATAN2(XVEL,YVEL)
PSIANGD=PSIANGR*RADIANS_2_DEGR !AIRCRAFT PATH ANGLE (DEG)

VU(1)=AC_VELOCITY ! A/C VEL VECTOR IN A/C COORDINATES
VU(2)=0.
VU(3)=0.

VELIN1=SQRT(XVEL*XVEL + YVEL*YVEL + ZVEL*ZVEL)
CONR=((PI**5./XLAM**4.)*KW**2.)*1.E-18 !Reflectivity constant for rain
! see Doviak & Zrnic, page 58

RCVR_BANDWIDTH=1./PWIDTH
RCVR_NOISE=10.**(RCVR_NOISE_DB/10.)

!
XTO= 290 DEG KELVIN
POWER_FROM_NOISE=RCVR_NOISE*XTO*RCVR_BANDWIDTH*BOLTZMAN_CONSTANT
SDNSE2=SQRT(POWER_FROM_NOISE/2.)
SDNSE=SQRT(POWER_FROM_NOISE) !NOISE STD DEVIATION
BINSZ=SAMPLE_INTERVAL*CC/2.
RANGES_PER_SUM=INT(PWIDTHM/DEL_RANGE + .1)
VMAX=CC/(4.*FREQ*PULSE_INTERVAL) ! NYQUIST VELOCITY
FC=AC_VELOCITY/(AG*BINSZ) ! CONSTANT FOR HAZARD FACTOR CALC.
CF=4.*PI*PULSE_INTERVAL*FILFAC/XLAM ! CONSTANT FOR CLUTTER FILTER
IF(CF .GE. 1.)WRITE(*,*) 'ERROR IN FILTER FACTOR'
ZZDBA=-50. ! INITIALIZE DBZ VALUE IN DBZ

CALL FILCO(VMAX,FILFAC) !calculate coefficients for EFILTER subroutine

CALL READANT ! read actual antenna pattern data from file WXANT.DAT

IF(DO_DISCRETES_FLAG .EQ. 1)CALL RDDISC2 ! read discrete target file

C read in weather databases
CALL READ_DATABASES

C calculate beamwidth
CALL GETBW(ANTENNA_TYPE,ANTENNA_RADIUS,APERTURE_TAPER, BMWID)
BMWID2=BMWID/2.

CALL WRITE_FILEHEADERS

```

```

CALL CALC_ROTATION_MATRICES

C      WRITE(*,*) 'READY TO START - PRESS ANY KEY', NUM_SCANS
C
C*****START AIRCRAFT MOTION LOOP*****
C
      IELKNT=0
      ISCAN_KNT=1
      DO 1998 SCAN_INDEX=1, NUM_SCANS

          NDIR=1
          IF(AZ.GT.AZSCAN_RANGE/2.) NDIR=-1
C
C*****DO ELEVATION BARS*****
C
      IELKNT=IELKNT + 1 ! ELEVATION SCAN COUNTER
      EL=ELSCAN_CENTER + (IELKNT-1)*DEL_ELSCAN

      IF (IELKNT.GT.NBARS) THEN
          IELKNT=0
          EL=ELSCAN_CENTER
          TIME_SECS=TIME_SECS + DEL_TIME
          ISCAN_KNT=ISCAN_KNT + 1 !START ANOTHER SCAN SEQUENCE
      END IF

C      UPDATE AIRCRAFT POSITION FOR NEW SCAN SEQUENCE

      XO=  XOFF  + XVEL*TIME_SECS  !X AXIS IS OFFSET
      YO =  YOFF  + YVEL*TIME_SECS  !Y AXIS USUALLY FLIGHT LINE
      ZO = -(ZOFF + ZVEL*TIME_SECS) !Z AXIS IS DOWN
      RO =  SQRT(XO*XO + YO*YO + ZO*ZO)

      IF(ZO.GT.0.)GO TO 1998          !A/C CRASHED
      BX=XO      ! SET OFFSETS IN DATA SETS
      BY=YO

      CALL VEL_WIND(XO, YO, ZO, XVEL, YVEL, AVR_VX, AVR_VY, AVR_WIND)
      AC_AIRSPEED=AC_VELOCITY - AVR_WIND
      AC_HEADING=ATAN2(XVEL, YVEL) *RADIANS_2_DEGR
      CALL ANG360(AC_HEADING)
C      WIND_DIRECTION=ATAN2(AVR_VX, AVR_VY) *RADIANS_2_DEGR
C      CALL ANG360(WIND_DIRECTION)
C
C*****START AZ SCAN LOOP*****
C
      AZ=AZSCAN_CENTER

      LINE_INDEX=0
      DO 544 AZSCAN_INDEX=1, NLINE
          LINE_KNT=LINE_KNT + 1
          TIME_SECS=TIME_SECS + SEC_PER_LINE

          IF (NDIR.EQ.1) THEN
              LINE_INDEX=AZSCAN_INDEX
          ELSE
              LINE_INDEX=NLINE-AZSCAN_INDEX + 1
          END IF
      END DO
  
```

```

!           AZIMUTH IN WEATHER DATA COORDINATES

AZR = AZSCAN_CENTER - AZSCAN_RANGE +
2           (LINE_INDEX-1)*DEL_AZSCAN + PSIANGD

IF(NLINE .EQ. 1)AZ=AZSCAN_CENTER

c           create rotation operator matices
CALL ROT(3,AZR,R5)
CALL ROT(2,EL,R6)

c           multiply matrix R6 by R5 to give R7
CALL GMPRD(R6,R5,R7,3,3,3)

c           multiply matrix R7 by R4 to give RFIN
CALL GMPRD(R7,R4,RFIN,3,3,3)

AZ=AZR-PSIANGD
IF(AZ.GT.180) AZ=AZ-360.
IF(AZ.LT.-180) AZ=AZ+360.

c           create rotation operator matices
CALL ROT(2,90.,R1)
CALL ROT(3,90.,R2)

c           multiply matrix R2 by R1 to give R3
CALL GMPRD(R2,R1,R3,3,3,3)

c           multiply matrix R3 by RFIN to give RFIN2
CALL GMPRD(R3,RFIN,RFIN2,3,3,3)

c           multiply matrix RV by VU to give VX
CALL GMPRD(RV,VU,VX,3,3,1)
C
C ***** GET ANTENNA ANGLE WITH RESPECT TO HORIZON *****
C
C           RFIN is conversion matrix for converting from weather
c           coordinates to radar antenna coordinates
C           VP is in rectangular coord. and RPS is spherical coord.

CALL HORIZ(RFIN,VP,RPS)

THTST=90.-RPS(2)-THETA_MAX ! starting theta angle
PHI1=RPS(3) - PHI_MAX
PHI2=RPS(3) + PHI_MAX

C           Subtract aircraft velocity if desired

IF(SUBTRACT_AC_velocity.EQ.1)THEN
    VELIN=VELIN1*SIN(RPS(2)*DEGREES_2_RAD)
ELSE
    VELIN=0.
END IF

C           Initialize velocity matrix

```

```

DO 320 III=1,2
  DO 321 JJJ=1,3
    VSAVE(JJJ,III)=0.
321   CONTINUE
320   CONTINUE

C     write line dependent file headers

      CALL WRTFLG(ISSCAN,IFSCAN,ISLINE,IFLINE,SCAN_INDEX,
2         LINE_INDEX,IQCODE,IFLG)
      CALL WRTFLG(ISSCAN,IFSCAN,ISLINE,IFLINE,SCAN_INDEX,
2         LINE_INDEX,IALMCD,IFLG2)

      IF(IFLG2.EQ.1)THEN
        CALL FILL_IQ_LINEHEADER
        WRITE(14)LHEAD
        XX=X
      END IF

c     CALL WRTFLG(ISSCAN,IFSCAN,ISLINE,IFLINE,SCAN_INDEX,
c     2         LINE_INDEX,IALMCD,IFLG)
c     IF(IFLG .EQ. 1)WRITE(12)SCAN_INDEX,LINE_INDEX,RO,AZ,EL

      CALL WRTFLG(ISSCAN,IFSCAN,ISLINE,IFLINE,SCAN_INDEX,
2         LINE_INDEX,ICSVCD,IFLG)
1388  IF(IFLG .EQ. 1)WRITE(13,1388)SCAN_INDEX,LINE_INDEX,RO,AZ,EL
      FORMAT(1X,I4,1H,,I4,1H,,F8.2,1H,,F8.2,1H,,F8.2)
C
C*****START RANGE BIN LOOP - BIN_NUMBER INDEX*****
C
      DO 888 BIN_NUMBER=1,NBINS
        RANGE=INITIAL_RANGE + (BIN_NUMBER-1)*BINSZ/XKMTM
        RRGC=RANGE + BINSZ/(2.*XKMTM)      ! RANGE TO CENTER OF BIN
        WRITE(*,458)BIN_NUMBER,RANGE,AZ,EL,SCAN_INDEX,ZZDBA
C
458      FORMAT(1X,
2      'BIN = ',I4,' RG = ',F5.1,' AZ = ',F5.1,
3      ' EL = ',F5.1,' SCAN = ',I4,' DBZ = ',F7.1)

c     calculate "True" velocity component along projected
c     antenna beam center line
      CALL VELTRU(RFIN,VX,VELIN,RO,RRGC,ZO,BX,BY,FVER,VTRU,VVER,
&                                     ANGM)
C
C *****RANGE CALCULATIONS*****
C

      RRNG=RRGC*XKMTM
      SRMIN=RRNG - PWIDTHM/2.
      SRMAX=RRNG + PWIDTHM/2

      TDELAY=2.*RRNG/CC
      DEL_RANGE=(SRMAX-SRMIN)/RANGES_PER_SUM
      PCONST=PWRTX*XLAM*XLAM/((4.*PI)**3.) ! Constants in Radar equation
      CP=PCONST/(RRNG**4.*RCVR_LOSS)      ! for discrete target

      K=0

```

```

        KK=0
        KNTZ=0
C
C*****WRITE HEADERS FOR BIN DEPENDENT DATA*****
C
        CALL WRNFLG (ISSCAN, IFSCAN, ISLINE, IFLINE, SCAN_INDEX,
2              LINE_INDEX, IQCODE, IFLG)
        CALL WRNFLG (ISSCAN, IFSCAN, ISLINE, IFLINE, SCAN_INDEX,
2              LINE_INDEX, IALMCD, IFLG2)

        IF (IFLG .EQ. 1) THEN
2          WRITE (21, 1389) SCAN_INDEX, LINE_INDEX, BIN_NUMBER, RO, RRNG,
              AZ, EL, XO, YO, ZO
        END IF

        IF (IFLG2.EQ.1) THEN
        CALL FILL_IQ_BINHEADER
        WRITE (14) BHEAD          !BINARY IQ FILE
        END IF

1389      FORMAT (1X, 3I5, 2F8.1, 2F7.1, 3F8.1)
C
        CALL WRNFLG (ISSCAN, IFSCAN, ISLINE, IFLINE, SCAN_INDEX,
2              LINE_INDEX, IFTCODE, IFLG)
        IF (IFLG .EQ. 1)
2          WRITE (23, 1389) SCAN_INDEX, LINE_INDEX, BIN_NUMBER, RO, RRNG,
3              AZ, EL, XO, YO, ZO

C ***** START RANGE, AZIMUTH & ELEVATION INTEGRATION LOOPS*****

        XLOSS=1.
        DO 577 MMM=1, 3
            RP (MMM) =VP (MMM) *RRNG
577      CONTINUE

c          calculate 2-way path loss (attenuation) due to rain

        IF (IATTN .EQ. 2) THEN
            CALL ATTN (RP, ZO, BINSZ, BX, BY, ANGM, XLOSS)
        END IF
        SUMZZ=0.

C
C*****START INTEGRATION LOOPS *****
        CALL INTEGRATION_LOOPS ()
C
C*****END INTEGRATION LOOPS*****
C
        KMAX=K
        IF (KNTZ .GT. 1 .AND. SUMZZ .GT. 0.) THEN
            ZZAVR=SUMZZ/KNTZ
            ZZDBA=10.*ALOG10 (ZZAVR)
        ELSE
            ZZDBA=DBZMIN
        ENDIF

        KKMAX=KK

```

```

CALL DISCRETE_TARGET_RETURN()

CALL SUM_UP_PULSES()

C
C*****END OF TIME AND SUM LOOPS*****
C
C Simulate effect of A/D converters in radar set on the signal
c ( no longer significant with 12-bit A/Ds)
  IF (NLEV.LT.2000) THEN
    CALL ADCONV (U1R,U2R, NPULSES, NPULSES, NLEV, GAIN_FACTOR, SDNSE, GN)
    CALL ADCONV (U1, U2, NPULSES, NPULSES, NLEV, GAIN_FACTOR, SDNSE, GN)
  END IF

CALL FILTER_IQ_SIGNALS()

CALL CALC_AVG_POWER()

CALL SIGNAL_PROCESSING()

C
C*****
C ***** WRITE OUTPUT DATA FILES *****
C*****
  RRG2=RRNG/XKMTM
  FTOL=FVER+FTRU

c Dummy variables used as place holders
  S22=0.
  S23=0.
  S24=0.

c copy values to 2-dim output array (OV)
CALL SETVAR (RRG2, VPP, VSP, VTRU, VVER, PRNS, PRRN, PRCL, PRCP, PDIS,
2 PTOL, GNDB, ZZDBA, WPP, FPP, FSP, VPPR, FTRU, FVER, FTOL, WPPR, S22,
3 S23, S24, BIN_NUMBER, OV)

C Reorder results of FFT so zero is at center (usually line 63)
DO 889 I=1, NPULSES
  IF (I.LE.NPULSES/2) II=I+NPULSES/2
  IF (I.GT.NPULSES/2) II=I-NPULSES/2
  IDATA (II) =IMAGDB (I)
  IF (IFTCODE.NE.0) IDATA2 (II) =IMAGDB2 (I)
889 CONTINUE
C
CALL WRNFLG (ISSCAN, IFSCAN, ISLINE, IFLINE, SCAN_INDEX, LINE_INDEX,
2 IFTCODE, IFLG)
IF (IFLG .NE. 0) THEN
  DELF=1./ (NPULSES*PULSE_INTERVAL)
  DELV=DELF*CC/ (2.*FREQ)
  WRITE (*, *) 'WRITING FFT OUTPUT FILE'
  DO 989 I=1, NPULSES
    OUT1=+ (I-1) *DELV - (NPULSES/2) *DELV
    OUT2=FLOAT (IDATA (NPULSES-I+1) )
    OUT3=FLOAT (IDATA2 (NPULSES-I+1) )
    WRITE (23, 1399) OUT1, OUT2, OUT3
989 CONTINUE
ENDIF

```

```

1399  FORMAT(1X,3f10.2)
990   CONTINUE
C
C     ***** WRITE IQ DATA FILE *****
C
      CALL WRTFLG(ISSCAN,IFSCAN,ISLINE,IFLINE,SCAN_INDEX,LINE_INDEX,
2          IQCODE,IFLG)
      CALL WRTFLG(ISSCAN,IFSCAN,ISLINE,IFLINE,SCAN_INDEX,
2          LINE_INDEX,IALMCD,IFLG2)

      IF(IFLG.NE.0)THEN
C
          WRITE(*,*)'WRITING IQ DATA OUTPUT FILE'

          DO I=1,NPULSES
              WRITE(21,1390) U1R(I),U2R(I),U1(I),U2(I)
          END DO
      END IF

      IF(IFLG2.NE.0)THEN
          CALL WRITE_IQ_DATA(NPULSES)
      END IF

1390  FORMAT(1X,4E12.3)
C
C     *****
C     save velocity values from this bin for Hazard factor calculations
      DO 322 III=1,3
          VSAVE(III,2)=VSAVE(III,1)
322   CONTINUE

          VSAVE(1,1)=VPP    ! velocity from pulse-pair processing
          VSAVE(2,1)=VSP    ! velocity from spectral averaging
          VSAVE(3,1)=VTRU   ! true velocity
C
      888   CONTINUE
C*****END OF RANGE BIN LOOP*****
C*****
C
      Average Hazard Factor over several range bins
      CALL AVRHAZ(OV,NHAVR,NBINS)
      CONHF=-FC*BINSZ
      CALL HAZLES(OV,2,17,NBINS,BINSZ,CONHF,RES,THRES)
C
C*****OUTPUT DATA TO FILES*****
C
      NWRITE=21    ! NUMBER OF VARIABLES TO WRITE TO FILE
      CALL WRTFLG(ISSCAN,IFSCAN,ISLINE,IFLINE,SCAN_INDEX,LINE_INDEX,
C          IALMCD,IFLG1)
      CALL WRTFLG(ISSCAN,IFSCAN,ISLINE,IFLINE,SCAN_INDEX,LINE_INDEX,
C          ICSVCD,IFLG2)
      IF(IFLG2 .EQ. 1)
2          WRITE(13,85) ((OV(K,MMM),K=1,NWRITE),MMM=1,NBINS)
C          IF(IFLG1 .EQ. 1)
C          2          WRITE(12) ((OV(K,MMM),K=1,NWRITE),MMM=1,NBINS)
C

```

```
85   FORMAT(1X,F7.2,1H,,F7.2,1H,,F7.2,1H,,F7.2,1H,,F7.2,
2     1H,,F7.2,1H,,F7.2,1H,,F7.2,1H,,F7.2,1H,,F7.2,1H,,F7.2,
3     1H,,F7.2,1H,,F7.2,1H,,F7.2,1H,,F7.3,
4     1H,,F7.3,1H,,F7.3,1H,,F7.3,1H,,F7.3,1H,,F7.3,1H,,F7.2)
```

```
C
C
```

```
544   CONTINUE
C*****END OF AZ SCAN LOOP-544*****
```

```
C
```

```
1998   CONTINUE
C*****END OF A/C MOTION LOOP-1998*****
```

```
C
```

```
2001   WRITE(*,*) 'PROGRAM COMPLETED'
C*****
```

```
CONTAINS
```

```
C
```

```
C *****
C *****
```

```
C   NAME:      READ_FILENAMES()
```

```
C
```

```
C   PURPOSE:  Read the file containing the names of files to be used
C             for input and output processing.
```

```
C
```

```
C   INPUT:    fn_file - file containing names to read
```

```
C
```

```
C   OUTPUT:   I/O file names
```

```
C
```

```
C *****
C *****
```

```
      SUBROUTINE READ_FILENAMES
```

```
      OPEN(7,FILE=fn_file,STATUS='OLD')      ! open file with input filenames
```

```
      READ(7,3333)UFILE      !u velocity database
      READ(7,3333)VFILE      !v velocity database
      READ(7,3333)WFILE      !w velocity database
      READ(7,3333)RFILE      !reflectivity database
      READ(7,3333)TKE_FILE    !Sigma wind database
      READ(7,3333)DFILE      !discrete target file
      READ(7,3333)OF1        !user name for output file #1
      READ(7,3333)OF2        !user name for output file #2
      READ(7,3333)OF3        !user name for output file #3
      READ(7,3333)OF4        !user name for output file #4
```

```
3333   FORMAT(A40)
```

```
      CLOSE(7)
```

```
C
```

```
      END SUBROUTINE READ_FILENAMES
```

```
C
```

```
C *****
C *****
```

```
C   NAME:      OPEN_FILES()
```

```
C
```

```
C   PURPOSE:  opens input and output data files
```

```
C
```

```
C   INPUT:    none
```

```

C
C     OUTPUT:  none
C
C*****
C*****
C     SUBROUTINE OPEN_FILES

C     OPEN(9, FILE=UFILE, FORM='FORMATTED',
C     .      access='sequential', STATUS='OLD')
C     OPEN(10, FILE=WFILE, FORM='FORMATTED',
C     .      access='sequential', STATUS='OLD')
C     OPEN(19, FILE=VFILE, FORM='FORMATTED',
C     .      access='sequential', STATUS='OLD')
C     OPEN(11, FILE=RFILE, FORM='FORMATTED',
C     .      access='sequential', STATUS='OLD')
C     IF (USE_TKE_FLAG.EQ.1) THEN
C     .      OPEN(30, FILE=TKE_FILE, FORM='FORMATTED',
C     .      .      access='sequential', STATUS='OLD')
C     END IF

c     OPEN(12, FILE=OF1, FORM='UNFORMATTED', STATUS='unknown')

C     IF (ICSVCD.NE.0) OPEN(13, FILE=OF2, STATUS='unknown')
C     IF (IQCODE.NE.0) OPEN(21, FILE=OF3, FORM='FORMATTED'
2     .      , STATUS='unknown')
C     IF (IFTCODE.NE.0) OPEN(23, FILE=OF4, FORM='FORMATTED'
2     .      , STATUS='unknown')
C     OPEN(16, FILE='WXANT.DAT', FORM='FORMATTED', STATUS='OLD')
C     OPEN(17, FILE=DFILE, STATUS='OLD')

C     IF (IALMCD.NE.0) OPEN(14, FILE=OF1, FORM='UNFORMATTED'
2     .      , STATUS='UNKNOWN')

C     END SUBROUTINE OPEN_FILES

C
C *****
C *****
C     NAME:      READ_PARAMETER_FILE()
C
C     PURPOSE:  Read the configuration parameters for this simulation
C     .      from the input file, convert to units used internally,
C     .      and store them.
C
C     INPUT:
C
C     OUTPUT:
C
C*****
C*****
C     SUBROUTINE READ_PARAMETER_FILE

C     *****AIRCRAFT MOTION*****
C
C     XOFF= IP_GET(1,1)*XKMTM      ! a/c x offset from weather origin
C     YOFF= IP_GET(2,1)*XKMTM      ! a/c y offset from weather origin

```

```

ZOFF= IP_GET(3,1)*FT_2_METERS*1000. !altitude relative to weather origin
XVEL= IP_GET(4,1)*XKTTMS           ! a/c velocity in x direction
YVEL= IP_GET(5,1)*XKTTMS           ! a/c velocity in y direction
ZVEL= IP_GET(6,1)*FT_2_METERS      ! a/c rate of climb
ROLL = IP_GET(7,1)                  ! roll, degrees
PITCH = IP_GET(8,1)                 ! pitch, degrees
YAW = IP_GET(9,1)                   ! yaw, degrees
WEIGHT=IP_GET(10,1) *1000.         ! A/C Weight in lbs
C
C *****SIMULATION PARAMETERS*****
C
  THETA_MAX = IP_GET(1,2)            ! max Theta = (AZ integration range)/2
  DEL_THETA = IP_GET(2,2)            ! delta theta: AZ integr. increment
  PHI_MAX   = IP_GET(3,2)            ! max phi = (El integration range)/2
  DEL_PHI   = IP_GET(4,2)            ! delta phi: El integr. increment
  DEL_RANGE = IP_GET(5,2)            ! delta range: Range integr. increment
  UNIFORM_RANDOM_NUM = IP_GET(6,2)  ! random number seed
  SUBTRACT_AC_VELOCITY= INT(IP_GET(7,2)) ! flag to subtract a/c velocity
  NUM_SCANS  = INT(IP_GET(8,2))      ! number of scans
  DEL_TIME   = IP_GET(9,2)           ! time between scans
  IDIAG     = INT(IP_GET(10,2))      ! enable diagnostic write statements
C
C *****WEATHER AND CLUTTER*****
C
  VEL_STD_DEV = IP_GET(1,3)          ! standard deviation of rain velocity
  CLUTTER_STD_DEV = IP_GET(2,3)      ! standard deviation of clutter
  DO_CLUTTER_FLAG = INT(IP_GET(3,3)) ! enable clutter calculations
  DO_DISCRETES_FLAG = INT(IP_GET(4,3)) ! enable discrete calculations
  DO_DBZ_FLAG = INT(IP_GET(5,3))     ! enable reflectivity calculations
  ZTHRES = IP_GET(6,3)               ! reflectivity calculation threshold
  DBZMIN = IP_GET(7,3)               ! minimum reflectivity value
  CLUTTER_TYPE = IP_GET(8,3)          ! ground clutter type
  SURFSD = IP_GET(9,3)               ! standard deviation of surface
  USE_TKE_FLAG = INT(IP_GET(10,3))   !flag to use TKE data
  ADD_DBZ = IP_GET(11,3)             !TKE offset
C
C *****RADAR PARAMETERS*****
C
  INITIAL_RANGE = IP_GET(1,4)         ! initial radar range, km
  NBINS = INT(IP_GET(2,4))            ! number of range bins
  PWRTX = IP_GET(3,4)                 ! transmitted power, watts
  RADAR_FREQ_GHZ = IP_GET(4,4)        ! radar freq, GHz
  PULSE_WIDTH_USEC = IP_GET(5,4)      ! pulse width, usec
  PULSE_INTERVAL_USEC = IP_GET(6,4)   ! pulse interval, usec
  RCVR_NOISE_DB = IP_GET(7,4)         ! receiver noise, dB
  RCVR_LOSS_DB = IP_GET(8,4)          ! receiver losses, dB
  PHASE_ERROR = IP_GET(9,4)           ! transmitter phase jitter, degrees
  FREQ_ERROR = IP_GET(10,4)           ! transmitter frequency jitter, Hz
  SAMPLE_INTERVAL_USEC=IP_GET(11,4)   ! pulse sample interval, usec
C
C *****ANTENNA AND SCAN*****
C
  ANTENNA_TYPE = INT(IP_GET(1,5)) ! antenna type
  ANTENNA_RADIUS = IP_GET(2,5)     ! antenna radius, meters
  APERTURE_TAPER = IP_GET(3,5)     ! aperture taper parameter
  SCAN_TYPE = INT(IP_GET(4,5))      ! scan type
  SCNRATE = IP_GET(5,5)             ! scan rate, deg/sec

```

```

AZSCAN_CENTER = IP_GET(6,5)      ! Azimuth scan center, degrees
ELSCAN_CENTER = IP_GET(7,5)      ! Elevation scan center, degrees
NLINE         = INT(IP_GET(8,5))  ! number of scan lines
NBARS         = INT(IP_GET(9,5))  ! number of scan bars
DEL_AZSCAN    = IP_GET(10,5)     ! azimuth scan increment
DEL_ELSCAN    = IP_GET(11,5)     ! elevation scan increment

C
C *****SIGNAL PROCESSING*****
C
NPULSES       = INT(IP_GET(1,6))  ! number of pulses/scan line
NUM_AD_BITS   = IP_GET(2,6)       ! number A/D bits
GAIN_FACTOR   = IP_GET(3,6)       ! AGC gain factor
PTHRES        = IP_GET(4,6)       ! processing threshold, dB
XIFIL         = IP_GET(5,6)       ! clutter filter code
FILFAC        = IP_GET(6,6)       ! clutter filter velocity cutoff
NHAVR         = INT(IP_GET(7,6))  ! # of bins for F-factor Avg.
THRES         = IP_GET(8,6)       ! threshold for F-factor algorithm
ARORD         = IP_GET(9,6)       ! AR model order

C
C *****DATA PRODUCTS*****
C
IALMCD        = INT(IP_GET(1,7))  ! Display program data code
ICSVCD        = INT(IP_GET(2,7))  ! C.S.V. output code
IQCODE        = INT(IP_GET(3,7))  ! IQ data output code
IARCOEF       = INT(IP_GET(4,7))  ! AR coefficient code
IARSPEC       = INT(IP_GET(5,7))  ! AR spectra data code
IFTCODE       = INT(IP_GET(6,7))  ! Fourier Spectra data code
ISSCAN        = INT(IP_GET(7,7))  ! start scan number
IFSCAN        = INT(IP_GET(8,7))  ! finish scan number
ISLINE        = INT(IP_GET(9,7))  ! start line number
IFLINE        = INT(IP_GET(10,7)) ! finish line number

IFIL=INT(XIFIL)

END SUBROUTINE READ_PARAMETER_FILE

C
C *****
C *****
C NAME:      WRITE_FILEHEADERS ()
C
C PURPOSE:  write fileheaders for each output file
C
C INPUT:
C
C OUTPUT:
C
C *****
C *****
SUBROUTINE WRITE_FILEHEADERS

FNAME='OUTPUT FILE TO ALARM PROGRAM WILL BE APPROX.'
CALL FSIZE2(FNAME,IALMCD,86,NUM_SCANS,NLINE,
2      NBINS,1,ISSCAN,IFSCAN,ISLINE,IFLINE,NUM_SCANS_IN_OUTPUT,
3      NUM_LINES_IN_OUTPUT)
C      IF(IALMCD.GT.0)WRITE(12)NUM_SCANS_IN_OUTPUT,NUM_LINES_IN_OUTPUT,
C      2      NUM_SCANS,NLINE,NBINS,NPULSES,AC_VELOCITY,PULSE_RATE,

```

```

c      3      AZSCAN_RANGE, DEL_AZSCAN, BINSZ, DEL_TIME

      FNAME='C.S.V. OUTPUT (PLOT) FILE WILL BE APPROX.'
      CALL FSIZE2(FNAME, ICSVCD, 86, NUM_SCANS, NLINE,
2         NBINS, 1, ISSCAN, IFSCAN, ISLINE, IFLINE, NUM_SCANS_IN_OUTPUT,
3         NUM_LINES_IN_OUTPUT)
      IF(ICSVCD.GT.0)WRITE(13, 901)NUM_SCANS_IN_OUTPUT,
2         NUM_LINES_IN_OUTPUT, NUM_SCANS, NLINE, NBINS, NPULSES,
3         AC_VELOCITY, PULSE_RATE, AZSCAN_RANGE, DEL_AZSCAN, BINSZ,
4         DEL_TIME
901  FORMAT(1X, I5, 1H, , F8.1, 1H, ,
2         F8.3, 1H, , F8.1, 1H, , F8.2, 1H, , F8.2, 1H, , F8.2)

      FNAME='THIS RUN WILL GENERATE AN IQ FILE OF APPROX.'
      CALL FSIZE2(FNAME, IQCODE, 16, NUM_SCANS, NLINE, NBINS,
2         NPULSES, ISSCAN, IFSCAN, ISLINE, IFLINE, NUM_SCANS_IN_OUTPUT,
3         NUM_LINES_IN_OUTPUT)
      IF(IQCODE.GT.0)WRITE(21, 901)NUM_SCANS_IN_OUTPUT,
2         NUM_LINES_IN_OUTPUT, NUM_SCANS, NLINE, NBINS, NPULSES,
3         AC_VELOCITY, PULSE_RATE, AZSCAN_RANGE, DEL_AZSCAN, BINSZ,
4         DEL_TIME

      IF(IALMCD.GT.0)THEN
          CALL FILL_IQ_FILEHEADER
          WRITE(14)FHEAD          !IQ OUT
          CTEST=FHEAD.CLABEL      !FOR DEBUGGING
      END IF

      FNAME='THIS RUN WILL GENERATE AN FFT SPECTRA FILE OF APPROX.'
      CALL FSIZE2(FNAME, IFTCODE, 12, NUM_SCANS, NLINE,
2         NBINS, NPULSES, ISSCAN, IFSCAN, ISLINE, IFLINE,
3         NUM_SCANS_IN_OUTPUT, NUM_LINES_IN_OUTPUT)
      IF(IFTCODE.GT.0)WRITE(23, 901)NUM_SCANS_IN_OUTPUT,
2         NUM_LINES_IN_OUTPUT, NUM_SCANS, NLINE, NBINS, NPULSES,
3         AC_VELOCITY, PULSE_RATE, AZSCAN_RANGE, DEL_AZSCAN, BINSZ,
4         DEL_TIME

      END SUBROUTINE WRITE_FILEHEADERS

C
C *****
C *****
C      NAME:      CALC_ROTATION_MATRICES()
C
C      PURPOSE: Create matrices for converting from one coordinate
C                system to another.
C
C      INPUT:
C
C      OUTPUT:
C
C *****
C *****
C      SUBROUTINE CALC_ROTATION_MATRICES
C
C      WRITE(*,*) 'CALCULATING MATRICES'
c      create rotation operator matrices

```

```

CALL ROT(3,-90.,R3)
CALL ROT(1,180.,R1)

c multiply matrix R3 by R1 to give RT
CALL GMPRD(R3,R1,RT,3,3,3)

c create rotation operator matix
CALL ROT(2,-GLIDESLOPE,R1)

c multiply matrix R1 by RT to give R3
CALL GMPRD(R1,RT,R3,3,3,3)

c Transpose an NxM matrix (R3) to give MxN matrix (RV)
CALL GMTRA(R3,RV,3,3)

c create multiple rotation matrix with rotation thru Euler angles
CALL EMAT(ROLL,PITCH,YAW,RE)

c multiply matrix RE by R3 to give R4
CALL GMPRD(RE,R3,R4,3,3,3)
END SUBROUTINE CALC_ROTATION_MATRICES
C*****
C*****

SUBROUTINE INTEGRATION_LOOPS()

DO I=1,RANGES_PER_SUM !RANGE INCREMENTS

SLANT_RANGE=SRMIN + DEL_RANGE*(I-1) + DEL_RANGE/2.

C CHECK FOR A/C ALT. < SLANT RANGE.

DTR=DEGREES_2_RAD

IF (ABS(ZO) .LE. SLANT_RANGE) THEN
PSI=ASIN(ZO/SLANT_RANGE)
AREAC=SLANT_RANGE*DEL_RANGE*DEL_THETA*
2 DTR/COS(PSI)
ELSE
PSI=-PI/2
AREAC=0
END IF

PHIS_PER_SUM=INT((PHI2-PHI1)/DEL_PHI)
IF(PHIS_PER_SUM.LT.1) PHIS_PER_SUM = 1

CALL AZIMUTH_INCREMENTS()

END DO

END SUBROUTINE INTEGRATION_LOOPS
C
C *****
C *****
C NAME: AZIMUTH_INCREMENTS()
C
C PURPOSE:

```

```

C
C     INPUT:
C
C     OUTPUT:
C
C*****
C*****
      SUBROUTINE AZIMUTH_INCREMENTS()

      DO J=1,THETAS_PER_SUM

C         KK=KK+1

      ! Calculate Theta at center of cell
      CALL FIND_THETA()

      IF(DO_CLUTTER_FLAG.EQ.1.AND.AREAC.GT.0.) THEN

          KK=KK + 1

          IF(KK.GT.KKLIM) KK=KKLIM

          CALL CALC_ANTENNA_GAIN()

          CALL CLUTTER_SIGNAL_AMPLITUDE()

      !random phase term for clutter

          CALL STRNUM(UNIFORM_RANDOM_NUM)!get uniformly distr random
number 0:1
          AAA(KK)=PI2*UNIFORM_RANDOM_NUM

          CALL NORMAL(RANDOM_NUMBER) !get normally distr random number
          DCLUV=RANDOM_NUMBER*CLUTTER_STD_DEV !Add std deviation of
clutter
C
c     VVV is the rate of phase change of the ground clutter cell relative to
c     the antenna due to motion i.e.- swaying of trees, current of river

          VVV(KK)=(((VX(1)*RHO(1)+VX(2)*RHO(2)+VX(3)*RHO(3))
2             /SLANT_RANGE-VELIN) + DCLUV)*CU
          END IF !CLUTTER DONE

          CALL ELEVATION_INCREMENTS()

      END DO !END AZIMUTH LOOP

      END SUBROUTINE AZIMUTH_INCREMENTS

C
C *****
C *****
C     NAME:     FIND_THETA()
C
C     PURPOSE:
C
C     INPUT:

```

```

C
C      OUTPUT:
C
C*****
C*****
C      SUBROUTINE FIND_THETA()
C
C      Calculate vector (rho) to scatterer
C
C          THTR=(THTST+DEL_THETA*(J-1) + DEL_THETA/2.)*DEGREES_2_RAD
C          STHT=SIN (THTR)
C          CTHT=COS (THTR)
C          SRCPS=SLANT_RANGE*COS (PSI)
C          RHO (1)=SRCPS*STHT
C          RHO (2)=SRCPS*CTHT
C          RHO (3)=ZO
C
C      END SUBROUTINE FIND_THETA
C
C*****
C*****
C      NAME:      CLUTTER_SIGNAL_AMPLITUDE()
C
C      PURPOSE:  CALCULATE SCATTERER AMPLITUDE EE(KK)
C
C      INPUT:    KK = INDEX OF SCATTERER
C
C      OUTPUT:   ARRAY OF SCATTERED AMPLITUDES
C
C*****
C*****
C      SUBROUTINE CLUTTER_SIGNAL_AMPLITUDE()
C
C          SIGODB=-90.
C          SIGO=1.E-9
C          DEPANG=ABS (PSI)      !8/14/99  ADDED CLUTTER MODEL
C
C      calculate sigma zero from analytical ground clutter model
C      IF(DO_CLUTTER_FLAG .EQ. 1)THEN
C          CALL CLUT7(CLUTTER_TYPE,SURFSD,DEPANG,SIGO,SIGODB)
C      END IF
C
C      calculate 2-way path loss (attenuation) due to rain
C      IF(IATTN.EQ.1)THEN
C          CALL ATTEN(RHO,ZO,BINSZ,BX,BY,ANGM, XLOSS)
C      END IF
C
C          CSIGC=SIGO*AREAC*CP*XLOSS
C          SRSIGC=SQRT (CSIGC)
C          EE(KK)=SRSIGC*GAIN      ! Amplitude of signal from ground clutter
C
C      END SUBROUTINE CLUTTER_SIGNAL_AMPLITUDE
C
C*****
C*****
C      NAME:      ELEVATION_INCREMENTS()
C

```

```

C      PURPOSE: CALLS ROUTINES TO CALCULATE RETURNS FROM WEATHER
C
C      INPUT:   VARIOUS
C
C      OUTPUT:  VALUES OF INCREMENTAL SCATTERER AMPLITUDE AND PHASE
C*****
C*****
      SUBROUTINE ELEVATION_INCREMENTS()
      INTEGER LLL

          DO 30 L=1,PHIS_PER_SUM

!              FIND LOCATION OF INTEGRATION VECTOR
              LLL=L
              CALL INTEGRATION_VECTOR(LLL)

!              GET REFLECTIVITY VALUE
              DBZ_STATUS=GET_DBZ()
              IF (DBZ_STATUS.EQ. .FALSE.) CYCLE

              K=K+1

              IF(K.GT.KLIM)K=KLIM

!              DETERMINE ANTENNA GAIN
              CALL CALC_ANTENNA_GAIN()

!              CALCULATE RAIN AMPLITUDE
              CALL RAIN_SIGNAL_AMPLITUDE()

!              PHASE TERM DUE TO WIND/RAIN VELOCITY
              CALL WEATHER_PHASE_TERM()

!              RANDOM PHASE TERM FOR RAIN
              CALL STRNUM(UNIFORM_RANDOM_NUM)!get uniformly distr random
number
              AA(K)=UNIFORM_RANDOM_NUM*PI2 ! Totally random phase term for
rain

          30      CONTINUE      !END ELEVATION LOOP
              XX=X

      END SUBROUTINE ELEVATION_INCREMENTS
C
C *****
C *****
C      NAME:   INTEGRATION_VECTOR()
C
C      PURPOSE: FIND LOCATION OF SCATTERER IN WEATHER DATA BASE
C
C      INPUT:   INDEX L
C
C      OUTPUT:  VARIOUS
C
C*****
C*****

```

```

SUBROUTINE INTEGRATION_VECTOR(LL)
INTEGER LL

      PHI=PHI2-(LL-1)*DEL_PHI - DEL_PHI/2.
      PHIR=PHI*DEGREES_2_RAD
      SPHI=SIN(PHIR)
      CPHI=COS(PHIR)
      RHO(1)=SLANT_RANGE*STHT*SPHI
      RHO(2)=SLANT_RANGE*CTHT*SPHI
      RHO(3)=SLANT_RANGE*CPHI
      AXX=RHO(1)
      AYY=RHO(2)
      AZZ=RHO(3)-ZO

      AXXX=RHO(1) + BX
      AYYY=RHO(2) + BY

      END SUBROUTINE INTEGRATION_VECTOR
C
C *****
C *****
C      NAME:      CALC_ANTENNA_GAIN()
C
C      PURPOSE:  CALCULATES GAIN OF RADAR ANTENNA
C
C      INPUT:    ANTENNA TYPE, VECTOR TO SCATTERER
C
C      OUTPUT:   ANTENNA GAIN
C
C *****
C *****
      SUBROUTINE CALC_ANTENNA_GAIN

c      multiply matrix RFIN2 by RHO to give RHOA
      CALL GMPRD(RFIN2,RHO,RHOA,3,3,1)

c      convert vector RHOA from rectangular coordinates to spherical
      CALL SPHER(RHOA,RHOS)

      ANTANG=RHOS(3)
      ANTTHT=RHOS(2)

! ****      IF(ANTANG .GE. 85.)ANTANG = 85. FROM DISCRET SIGNAL AMPLITUDE ****

c      Calculate antenna gain based on antenna type
      IF(ANTENNA_TYPE .EQ. 1)
2          CALL ANTPAT(ANTENNA_RADIUS,APERTURE_TAPER,ANTANG,GAIN)
      IF(ANTENNA_TYPE .EQ. 2)
2          CALL ANTFLAT(ANTANG,GAIN)
      IF(ANTENNA_TYPE .EQ. 3)
2          CALL ANTREAL(ANTANG,ANTTHT,GAIN)

      END SUBROUTINE CALC_ANTENNA_GAIN

C *****
C *****
C      NAME:      GET_DBZ()

```

```

C
C   PURPOSE: GETS DBZ VALUE OF SCATTERER FROM WEATHER DATA BASE
C
C   INPUT:   LOCATION OF SCATTERER IN DATA BASE
C
C   OUTPUT:  DBZ VALUE, LOGICAL FALSE IF BELOW THRESHOLDS
C
C*****
C*****
C   LOGICAL FUNCTION  GET_DBZ ()
C
C           GET_DBZ = .TRUE.
C           DBZ=ZTHRES
C
C           CALL GET_WEATHER_PARAMETERS
2            (U,V,W,DBZ,DELTA_WIND,AXXX,AYYY,AZZ,3,ANGM)
C
C           IF(DBZ .LE. ZTHRES) GET_DBZ = .FALSE.
C           IF(DBZ .LE. DBZMIN) DBZ=DBZMIN
C
C           ZZZ=10**(DBZ/10.)
C
C   END FUNCTION GET_DBZ
C
C*****
C*****
C   NAME:     RAIN_SIGNAL_AMPLITUDE ()
C
C   PURPOSE:  CALCULATE AMPLITUDE OF RAIN SCATTERER
C
C   INPUT:    VARIOUS
C
C   OUTPUT:   ARRAY E(K) OF AMPLITUDES
C
C*****
C*****
C   SUBROUTINE RAIN_SIGNAL_AMPLITUDE ()
C   REAL ETA
C
C           calculate 2-way path loss (attenuation) due to rain
C           IF(IATTN.EQ.1) THEN
C               CALL ATTN(RHO,ZO,BINSZ,BX,BY,ANGM, XLOSS)
C           END IF
C
C           ETA=ZZZ*CONR      ! reflectivity of rain
C           TEST=ABS (ANTANG) -BMWID2
C           ZADD=0.
C
C           IF (TEST .LE. 0.) THEN
C               ZADD=ZZZ
C               KNTZ=1+KNTZ
C           ENDIF
C
C           SUMZZ=SUMZZ + ZADD
C
C           AREA=SLANT_RANGE*DEL_RANGE*DEL_THETA*DEGREES_2_RAD
C           CVOL=AREA*SLANT_RANGE*DEL_PHI*DEGREES_2_RAD

```

```

                CSIG=ETA*CVOL*CP*XLOSS
                SRSIG=SQRT(CSIG)
                E(K)=SRSIG*GAIN          ! Amplitude of signal from rain scatterer

        END SUBROUTINE RAIN_SIGNAL_AMPLITUDE
C
C *****
C *****
C      NAME:      WEATHER_PHASE_TERM()
C
C      PURPOSE:  CALCULATE DOPPLER VELOCITY OF A SCATTERER
C                FROM VELOCITY AND LINE-OF-SIGHT VECTORS
C
C      INPUT:    VECTORS, VARIOUS CONSTANTS, SCATTERER INDEX
C
C      OUTPUT:   ARRAY OF DOPPLER VELOCITY TERMS FOR SCATTERERS
C
C *****
C *****
        SUBROUTINE WEATHER_PHASE_TERM()
C
C      DETERMINE U, V, & W WIND VELOCITIES
C
C      CALL GET_WEATHER_PARAMETERS
2          (U,V,W,DBZ,DELTA_WIND,AXXX,AYYY,AZZ,2,ANGM)
C
C      IF(USE_TKE_FLAG.EQ.1) THEN
C          DV=DELTA_WIND
C      ELSE
C          CALL NORMAL(RANDOM_NUMBER) ! get distr random number
C          DV=RANDOM_NUMBER*VEL_STD_DEV
C      END IF
C
C      VXX=VX(1)-U
C      VYY=VX(2)-V
C      VZZ=VX(3)-W
C
C      VV is the rate of phase change of weather (rain)
C      due to relative motion along L.O.S.
C
C      VLOS=((VXX*RHO(1)+VYY*RHO(2)+VZZ*RHO(3)) !LOS VELOCITY
2          /SLANT_RANGE -VELIN)
C
C      VV(K)=VLOS*CU ! DOPPLER FREQ.
C
C      VV(K)=VV(K) + DV*CU ! ADD RANDOM DV TO DOPPLER
C
        END SUBROUTINE WEATHER_PHASE_TERM
C
C *****
C *****
C      NAME:      DISCRETE_TARGET_RETURN()
C
C      PURPOSE:  GETS RADAR RETURN FROM DISCRETE TARGETS IF USED
C
C      INPUT:
C
C

```

```

C      OUTPUT:
C
C*****
C*****
      SUBROUTINE DISCRETE_TARGET_RETURN()
      KKD=0

      DO 1868 KS=1,NSEG

          IF(DO_DISCRETES_FLAG .NE. 1) EXIT
          KTMAX=NND(KS)
          KKS=KS
          CALL DISCRETE_SIGNAL_AMPLITUDE(KKS)

1868      CONTINUE

      KDMAX=KKD

      END SUBROUTINE DISCRETE_TARGET_RETURN
C
C*****
C*****
C      NAME:      DISCRETE_SIGNAL_AMPLITUDE()
C
C      PURPOSE:  CALCULATES AMPLITUDE OF DISCRETE TARGET RETURN
C
C      INPUT:
C
C      OUTPUT:
C
C*****
C*****
      SUBROUTINE DISCRETE_SIGNAL_AMPLITUDE(KSS)
      INTEGER KSS

      DO 1867 KT=1,KTMAX

C      ***** GET ANTENNA ANGLE *****

          RHO(2)=YYD(KSS,KT) - YO
          RHO(3)=UPD(KSS,KT) + ZO
          RHO(1)=XXD(KSS,KT) - XO
          RHO1SQ=RHO(1)*RHO(1)
          RHO2SQ=RHO(2)*RHO(2)
          RHO3SQ=RHO(3)*RHO(3)
          RGD=SQRT(RHO1SQ+RHO2SQ+RHO3SQ)
          IF(RGD.LT.SRMIN.OR.RGD.GT.SRMAX) CYCLE

          KKD=KKD+1
          IF(KKD.GT.KDLIM)KKD=KDLIM

          CALL CALC_ANTENNA_GAIN()

C      ***** RAW SIGNAL *****

          SIGO=10.** (RCSD(KSS,KT)/10.)

```

```

c      ***** calculate 2-way path loss (attenuation) due to rain ***
      IF (IATTN.EQ.1)
2        CALL ATTEN (RHO, ZO, BINSZ, BX, BY, ANGM, XLOSS)

      CSIGD=SIGO*CP*XLOSS
      SRSIGD=SQRT (CSIGD)
      EED(KKD)=SRSIGD*GAIN          ! Amplitude of signal from discrete target

      CALL DISCRETE_PHASE_TERM(KSS)

1867  CONTINUE

      END SUBROUTINE DISCRETE_SIGNAL_AMPLITUDE

C
C *****
C *****
C      NAME:      DISCRETE_PHASE_TERM()
C
C      PURPOSE:  CALCULATES DOPPLER TERM FOR DISCRETE TARGETS
C
C      INPUT:
C
C      OUTPUT:
C
C *****
C *****
      SUBROUTINE DISCRETE_PHASE_TERM(KSS)
      INTEGER KSS

!      totally random phase term for discrettes
      CALL STRNUM (UNIFORM_RANDOM_NUM) !get uniformly distr random number 0:1
      AAD(KKD)=PI2*UNIFORM_RANDOM_NUM

      VXX=VX (1) +VXD (KSS, KT)
      VYY=VX (2) +VYD (KSS, KT)
      VZZ=VX (3)

C      VVD is the rate of phase change of discrete targets relative to the
antenna
c      due to motion of targets along L.O.S.

      VLOS= ( ( VXX*RHO (1) +VYY*RHO (2) +VZZ*RHO (3) ) /RGD-VELIN )
      VVD (KKD)=VLOS*CU

      END SUBROUTINE DISCRETE_PHASE_TERM

C
C *****
C *****
C      NAME:      SUM_UP_PULSES ()
C
C      PURPOSE:  ADD UP CONTRIBUTIONS FROM CLUTTER, RAIN & DISCRETES
C                FOR THIS BIN, THIS SCAN LINE
C
C      INPUT:
C
C      OUTPUT:
C
C *****

```

```

C*****
SUBROUTINE SUM_UP_PULSES()

DO 50 II=1,NPULSES

!      time between 1st pulse and current pulse for this scan line
TT= PULSE_INTERVAL*(II-1)

c      get random phase term due to phase jitter of transmitter
CALL NORMAL(RANDOM_NUMBER) ! get normally distr random number
PHASE_JITTER=RANDOM_NUMBER*PHASE_ERROR*DEGREES_2_RAD

c      get random phase term due to frequency jitter of transmitter
CALL NORMAL(RANDOM_NUMBER) ! get normally distr random number
FREQ_JITTER=RANDOM_NUMBER*FREQ_ERROR*PI2

c      combine jitter terms for transmitter
XMIT_JITTER=PHASE_JITTER + FREQ_JITTER*TDELAY

CALL SUM_RAIN_RETURN()

CALL SUM_CLUTTER_RETURN()

CALL SUM_DISCRETE_TARGET_RETURN()

CALL RECEIVER_NOISE()

CALL FORM_IQ_SIGNALS()

50    CONTINUE
END SUBROUTINE SUM_UP_PULSES

C
C *****
C *****
C      NAME:      SUM_RAIN_RETURN()
C
C      PURPOSE:  SUMS RAIN SCATTERER RETURNS TO GET A VALUE OF I & Q
C
C      INPUT:
C
C      OUTPUT:
C
C*****
C*****
SUBROUTINE SUM_RAIN_RETURN()

SUM1=0.
SUM2=0.
DO 60 KRN=1,KMAX
  PHASE= AA(KRN)+VV(KRN)*TT + XMIT_JITTER ! sum random phase terms
  XXX=E(KRN) !amplitude of radar signal from rain scatterer
  SUM1 = SUM1 + XXX*COS(PHASE) ! sum in-phase amplitudes of rain signal
  SUM2 = SUM2 + XXX*SIN(PHASE) ! sum quadrature amplitudes of rain signal
60  CONTINUE
PRAIN(II)=SUM1*SUM1 + SUM2*SUM2 ! total signal power from rain

IF(KMAX.GT.0) THEN !FOR DEBUGGING

```

```
XX=X
END IF
```

```
END SUBROUTINE SUM_RAIN_RETURN
```

```
C
C *****
C *****
C NAME: SUM_CLUTTER_RETURN()
C
C PURPOSE: SUMS CLUTTER SCATTERER RETURNS TO GET A VALUE OF I & Q
C
C INPUT:
C
C OUTPUT:
C
C *****
C *****
C SUBROUTINE SUM_CLUTTER_RETURN()
C
C SUM1C=0.
C SUM2C=0.
C DO 62 KKC=1, KKMAX
C PHASE= AAA(KKC)+VVV(KKC)*TT + XMIT_JITTER ! sum random phase terms
C XXX=EE(KKC) !amplitude of radar signal from clutter cell
C SUM1C = SUM1C + XXX*COS(PHASE) ! sum in-phase ampl. of clutter signals
C SUM2C = SUM2C + XXX*SIN(PHASE) ! sum quadrature amplitudes of clutter
62 CONTINUE
C
C PCLUT(II)=SUM1C*SUM1C + SUM2C*SUM2C ! signal power due to clutter
C CC1(II)=SUM1C ! reserve copy for later filtering
C CC2(II)=SUM2C ! "
C END SUBROUTINE SUM_CLUTTER_RETURN
```

```
C
C *****
C *****
C NAME: SUM_DISCRETE_TARGET_RETURN()
C
C PURPOSE: Calculate power in returned radar signal contributed by discrete
targets
C
C INPUT:
C
C OUTPUT:
C
C *****
C *****
C SUBROUTINE SUM_DISCRETE_TARGET_RETURN()
C
C SUM1D=0.
C SUM2D=0.
C PDISC(II)=0.
C IF(DO_DISCRETES_FLAG .NE. 1)RETURN
C DO 620 KKKD=1, KDMAX
C PHASE= AAD(KKKD)+VVD(KKKD)*TT + XMIT_JITTER ! sum random phase terms
C XXX=EED(KKKD) !amplitude of radar signal from discrete target
C SUM1D = SUM1D + XXX*COS(PHASE) ! sum in-phase amplitude of discretess
```

```

        SUM2D = SUM2D + XXX*SIN(PHASE) ! sum Q amplitude of discrete signal
620   CONTINUE
621   PDISC(II)=SUM1D*SUM1D + SUM2D*SUM2D

        END SUBROUTINE SUM_DISCRETE_TARGET_RETURN
C
C *****
C *****
C   NAME:      RECEIVER_NOISE()
C
C   PURPOSE: Calculate power in returned radar signal added by receiver noise
C   INPUT:
C
C   OUTPUT:
C
C *****
C *****
SUBROUTINE RECEIVER_NOISE()

CALL NORMAL(RANDOM_NUMBER) ! get normally distr random number
PN1=RANDOM_NUMBER*SDNSE2
SUM1=SUM1+PN1                ! Add noise to in-phase (I) rain signal

CALL NORMAL2(RANDOM_NUMBER) ! get normally distr random number
PN2=RANDOM_NUMBER*SDNSE2
SUM2=SUM2+PN2                ! Add noise to quadrature (Q) rain signal

PNSE(II)=PN1*PN1 + PN2*PN2 ! calculate power of receiver noise
END SUBROUTINE RECEIVER_NOISE

C
C *****
C *****
C   NAME:      FORM_IQ_SIGNALS()
C
C   PURPOSE: SUMS CLUTTER, RAIN, AND DISCRETE SIGNALS TO GET
C             SIMULATED I AND Q VALUES AND TOTAL POWER RECEIVED
C
C   INPUT:
C
C   OUTPUT:
C
C *****
C *****
SUBROUTINE FORM_IQ_SIGNALS()

!   Amplitude of signal from rain, clutter,discretes & receiver noise for I
& Q
U1(II)=SUM1+SUM1C+SUM1D      ! I signal
U2(II)=- (SUM2+SUM2C+SUM2D) ! Q signal

U1R(II)=U1(II)              ! Save a copy for filtering later
U2R(II)=U2(II)              ! "
PTOT(II)=U1(II)*U1(II) + U2(II)*U2(II) !total power for this pulse & bin

C   WRITE(*,*) II,U1(II),U2(II),PTOT(II)

```

```

END SUBROUTINE FORM_IQ_SIGNALS
C
C *****
C *****
C NAME: FILTER_IQ_SIGNALS()
C
C PURPOSE: HIGH-PASS TIME-DOMAIN CLUTTER FILTER FOR USE WHEN
C AIRCRAFT DOPPLER VELOCITY IS REMOVED.
C (I.E., AIRCRAFT DOPPLER IS SUBTRACTED IN RADAR)
C
C INPUT: FILTER TYPE (IFIL),
C
C OUTPUT:
C
C *****
C *****
SUBROUTINE FILTER_IQ_SIGNALS()
! high pass time-domain digital filter to suppress clutter
!
IF(IFIL .GT. 0)THEN
DO 8888 JJJ=1,IFIL
CALL DFILTER(CC1,CC2,CF,NPULSES) !CLUTTER ONLY
! WRITE(*,*) 'ONE STAGE CLUTTER FILTER COMPLETED'
CALL DFILTER(U1,U2,CF,NPULSES) !COMPOSITE SIGNAL
8888 CONTINUE
! 1st or 2nd degree Butterworth filter for suppression of ground clutter
ELSEIF(IFIL .LT. 0)THEN
CALL EFILTER(U1,U2,IFIL,NPULSES)
CALL EFILTER(CC1,CC2,IFIL,NPULSES)
!
! No filtering
ELSEIF(IFIL .EQ. 0)THEN
! WRITE(*,*) 'NO CLUTTER FILTER'
ENDIF
!
END SUBROUTINE FILTER_IQ_SIGNALS
C
C *****
C *****
C NAME: CALC_AVG_POWER()
C
C PURPOSE: CALCULATE AVERAGE POWER FROM CLUTTER, WEATHER, ETC.
C
C INPUT:
C
C OUTPUT:
C
C *****
C *****
SUBROUTINE CALC_AVG_POWER()
PSMALL=1.E-18
PRRN=PSMALL
PRCL=PSMALL
PDIS=PSMALL
PRNS=PSMALL

```

```

PTOL=PSMALL
PRCP=PSMALL
PTEST=PSMALL
DO 782 JJJ=1, NPULSES
    PRCP=CC1 (JJJ)**2 + CC2 (JJJ)**2 + PRCP
    PRRN=PRRN + PRIN (JJJ)          !RAIN
    PRCL=PRCL + PCLUT (JJJ)        !CLUTTER
    PRNS=PRNS + PNSE (JJJ)         !NOISE
    PDIS=PDIS + PDISC (JJJ)        !DISCRETES
    PTOL=PTOL + PTOT (JJJ)         !TOTAL
782 CONTINUE

PRES=PTOL-PRRN-PRCL
IF (PRES .LE. PTEST) PRES=PTEST
STHRES=PFAC*PRNS/NPULSES

c      convert quantities to DB for output
PRRN=10.*ALOG10 (PRRN/NPULSES)
PRCL=10.*ALOG10 (PRCL/NPULSES)
PRNS=10.*ALOG10 (PRNS/NPULSES)
PTOL=10.*ALOG10 (PTOL/NPULSES)
PRES=10.*ALOG10 (PRES/NPULSES)
PRCP=10.*ALOG10 (PRCP/NPULSES)
PDIS=10.*ALOG10 (PDIS/NPULSES)
C      GNDB=10.*ALOG10 (GN)
GNDB=999.

      END SUBROUTINE CALC_AVG_POWER
C
C *****
C *****
C      NAME:      SIGNAL_PROCESSING ()
C
C      PURPOSE:  CALCULATE VARIOUS FILTERED AND UNFILTERED MOMENTS
C                AND WINDSHEAR HAZARD FACTOR
C
C      INPUT:
C
C      OUTPUT:
C
C *****
C *****
      SUBROUTINE SIGNAL_PROCESSING ()

c      Calculate average Doppler velocity and spectral width using
c      pulse pair processing algorithms

C      FILTERED FOR CLUTTER:
CALL PPP (U1, U2, NPULSES, PULSE_INTERVAL, STHRES, VPP, WPP)

C      UNFILTERED:
CALL PPP (U1R, U2R, NPULSES, PULSE_INTERVAL, STHRES,
2          VPPR, WPPR)

c      Calculate average Doppler Velocity in range bin using Spectral Averaging
CALL WXANAL (U1, U2, NPULSES, PULSE_INTERVAL, STHRES,
2          IMAGDB, VSP)

```

```

        IF(IFTCODE .NE. 0) CALL WXANAL(U1R,U2R,NPULSES,PULSE_INTERVAL,
2          STHRES,IMAGDB2,DUM)

C      Calculates Hazard Index (F Factor) by differencing adjacent velocity
values
        IF(BIN_NUMBER.GE.3)CALL FFAC(FC,VSAVE,VPP,VSP,VTRU,FPP,FSP,FTRU)

        END SUBROUTINE SIGNAL_PROCESSING

C *****
C *****
C      NAME:      FILL_IQ_FILEHEADER
C
C      PURPOSE:  SETS VALUES IN FILEHEADER STRUCTURE FOR OUTPUT FILE
C
C      INPUT:
C
C      OUTPUT:
C
C *****
C *****

SUBROUTINE FILL_IQ_FILEHEADER

        ELSCAN_RANGE=2.*(NBARS-1)*ABS(DEL_ELSCAN)

        FHEAD%CLABEL          = 'ADWR'
        FHEAD%START_TIME_HR   =10          !USE FOR TESTS
        FHEAD%START_TIME_MIN  =0
        FHEAD%START_TIME_SEC  =0
        FHEAD%NUM_SCANS       =NUM_SCANS
        FHEAD%NUM_LINES       =NLINE
        FHEAD%NUM_SCANS_OUTPUT=NUM_SCANS_IN_OUTPUT
        FHEAD%NUM_LINES_OUTPUT=NUM_LINES_IN_OUTPUT
        FHEAD%NUM_RANGE_BINS  =NBINS
        FHEAD%NUM_PULSES      =NPULSES
        FHEAD%PULSE_REP_FREQ  =PULSE_RATE
        FHEAD%PULSE_WIDTH     =PWIDTHM      !METERS
        FHEAD%SCAN_CENTER     =AZSCAN_CENTER
        FHEAD%SCAN_RANGE      =AZSCAN_RANGE
        FHEAD%SCAN_INCREMENT  =DEL_AZSCAN
        FHEAD%TILT_CENTER     =ELSCAN_CENTER
        FHEAD%TILT_RANGE      =ELSCAN_RANGE
        FHEAD%TILT_INCREMENT  =DEL_ELSCAN
        FHEAD%SWEEP_SPEED     =SCNRATE      !DEG/SEC
        FHEAD%TOTAL_BARS      =NBARS
        FHEAD%RADAR_GATE_DELAY=300.0 !METERS
        FHEAD%BIN_WIDTH       =BINSZ !M
        FHEAD%AC_GND_SPEED    =AC_VELOCITY !M/S
        FHEAD%RADAR_FREQ      =FREQ        !HZ
        FHEAD%AC_WEIGHT       =WEIGHT !POUNDS

        END SUBROUTINE

C *****
C *****

```

```

C *****
C   NAME:      FILL_IQ_LINEHEADER
C
C   PURPOSE:  SETS VALUES IN LINEHEADER STRUCTURE FOR OUTPUT FILE
C
C   INPUT:
C
C   OUTPUT:
C
C *****
C *****
C *****

```

```

SUBROUTINE FILL_IQ_LINEHEADER

```

```

    XMIN=0.
    XSEC=0.
    XHR=0.
    IF (TIME_SECS.GT.60.) THEN
        XMIN=MOD (TIME_SECS,60.)
        XSEC=TIME_SECS - XMIN*60.
        IF (XMIN.GT.60.) THEN
            XHR=MOD (XMIN,60.)
            XMIN=XMIN - XHR*60.
        END IF
    ELSE
        XSEC=TIME_SECS
    END IF

    LHEAD%LINELABEL           = 'LINE'
    LHEAD%TIME_SEC            = INT (XSEC)           !SECS
    LHEAD%TIME_MIN            = INT (XMIN)
    LHEAD%TIME_HR             = INT (XHR)
    LHEAD%SCAN_INDEX          =SCAN_INDEX
    LHEAD%LINE_INDEX          =LINE_INDEX
    LHEAD%STARTING_RANGE      =RO !METERS
    LHEAD%SCAN_ANGLE          =AZ !DEG
    LHEAD%TILT_ANGLE          =EL !DEG
    LHEAD%CURRENT_BAR         =IELKNT !CODE
    LHEAD%AC_XCOORD           =XO !METERS
    LHEAD%AC_YCOORD           =YO !METERS
    LHEAD%AC_ZCOORD           =ZO !METERS
    LHEAD%AC_LATITUDE         =0.0 !DEG (NOT AVAILABLE)
    LHEAD%AC_LONGITUDE        =0.0 !DEG
    LHEAD%AC_HEADING          =AC_HEADING !DEG
    LHEAD%AC_TRACK            =0.0 !DEG
    LHEAD%AC_TRUE_AIRSPEED    =AC_AIRSPEED !M/S

```

```

END SUBROUTINE

```

```

C *****
C *****
C *****
C   NAME:      FILL_IQ_BINHEADER
C
C   PURPOSE:  SETS VALUES IN BINHEADER STRUCTURE FOR OUTPUT FILE
C
C   INPUT:
C
C

```

```
C      OUTPUT:
C
C*****
C*****
C*****
```

```
      SUBROUTINE FILL_IQ_BINHEADER
```

```
          BHEAD%BIN_NUMBER      =BIN_NUMBER
          BHEAD%RANGE_TO_BIN    =RRNG  !METERS
```

```
      END SUBROUTINE
```

```
C*****
C *****
C *****
```

```
      NAME:      WRITE_IQ_DATA
```

```
      PURPOSE:  SETS IQ VALUES IN IQ STRUCTURE FOR OUTPUT FILE
                AND WRITES THEM TO FILE #14
```

```
      INPUT:
```

```
      OUTPUT:
```

```
C*****
C*****
C*****
```

```
      SUBROUTINE WRITE_IQ_DATA(IP)
```

```
          TYPE(IQ_TYPE_SIM) XIQ(IP)
```

```
          DO K=1,IP
              XIQ(K)%IDAT=U1R(K)
              XIQ(K)%QDAT=U2R(K)
          END DO
```

```
          WRITE(14)XIQ
```

```
      END SUBROUTINE
```

```
C*****
C*****
```

```
      END PROGRAM
```

```
C*****END OF MAIN PROGRAM*****
```

```
C
C *****
C *****
```

```
      NAME:      CLUT7 ()
```

```
      PURPOSE:  THEORETICAL GROUND CLUTTER MODEL (RTI TM DATED 6/27/99)
```

```
      INPUT:    DEPRESSION ANGLE "ANG" IN RADIANS
```

```
      OUTPUT:   SIGMA ZERO VALUE OF SCATTERER ON GROUND (DB)
```

```
C
```

C*****
C*****

```
      SUBROUTINE CLUT7 (ITYPE,SD,ANG, SIGO,SIGODB)
      USE CONST
      IMPLICIT INTEGER (I-N)
      DIMENSION AA(8),BB(8),CCC(8),DD(8)
C      COMMON/CONST/PI,DEGREES_2_RAD,RADIANS_2_DEGR,CC,FREQ,XLAM
      DATA AA/ .250, .023, .006, .002, 2.00, .025, .195, 00.0/
      DATA BB/ .830, 1.50, 1.50, .640, 1.80, 1.70, 1.70, 00.0/
      DATA CCC/.0013, .012, .012, .002, .015, .0016, .0016, 00.0/
      DATA DD/ 2.30, 0., 0., 0., 0., 0., 0., 0./

      AAAA=AA (ITYPE)
      BBBB=BB (ITYPE)
      C=CCC (ITYPE)
      D=DD (ITYPE)

      CK=0.
      CCEXP=1.
      ANGR=ABS (ANG)
      ANGD=ANG*RADIANS_2_DEGR
      IF (D.NE.0.) THEN
          CK=.1*SD/XLAM
          CCEXP=EXP(-D/(1. + CK))
      END IF

      SIGO=(AAAA* ((ANGR + C)**BBBB)) *CCEXP
      SIGODB=10.*ALOG10 (SIGO)
      RETURN
      END
```

FUNCTION inpars()

*-----

* Purpose:
* To manage input data.

*-----

```
REAL inpars
REAL ip_read
REAL ip_get

INTEGER lun
INTEGER segment
INTEGER irecord

REAL par(15,7)

SAVE par

REAL dum
INTEGER i
INTEGER imax
INTEGER j
```

```

INTEGER nrecord(7)

DATA nrecord
. /10,11,11,11,12,10,11/      !CHANGED 3RD TO 11

ENTRY ip_read(lun)

*-----*

* Purpose:

*   To fetch input parameters.

* Inputs:

*   lun - Logical unit number

* Outputs:

*   ip_read - Error indicator

*   par(01..15,1..7) - Input parameters

*       par(01..10,1) - "Aircraft" segment

c       par(01,1)      A/C X-Offset from WX Origin (km)
c       par(02,1)      A/C Y-Offset from WX Origin (km)
c       par(03,1)      A/C Altitude (1000 ft)
c       par(04,1)      A/C X-Velocity (kts)
c       par(05,1)      A/C Y-Velocity (kts)
c       par(06,1)      A/C Rate-of-Climb (ft/sec)
c       par(07,1)      Roll Attitude (deg)
c       par(08,1)      Pitch Attitude (deg)
c       par(09,1)      Yaw Attitude (deg)
c       par(10,1)      Weight (1000 lbs)

*       par(01..11,2) - "Simulation Environment" segment

*       par(01,2) - Az integration range/2 (deg)
*       par(02,2) - Az integration increment (deg)
*       par(03,2) - El integration range/2 (deg)
*       par(04,2) - El integration increment (deg)
*       par(05,2) - Rng integration increment (m)
*       par(06,2) - Random number seed (>=1)
*       par(07,2) - Subtract Aircraft Velocity (1=Y)
*       par(08,2) - Number of Complete Raster Scans
*       par(09,2) - Time Between Scans (secs)
*       par(10,2) - 2D/3D Weather files
*       par(11,2) - Spare

*       par(01..10,3) - "Weather and Clutter" segment

*       par(01,3) - Rain standard deviation (m/s)
*       par(02,3) - Clutter standard deviation (m/s)
*       par(03,3) - Clutter calc. flag (1 = ON, 0 = OFF)
*       par(04,3) - Discrete calc. flag (1 = ON,0 = OFF)
*       par(05,3) - Reflectivity calc. flag (1=on)

```

```

*      par(06,3) - Reflectivity calc. thres. (dBz)
*      par(07,3) - Minimum reflectivity (dBz)
*      par(08,3) - Ground Clutter Type Code
*      par(09,3) - Surface Standard Deviation (m)
*      par(10,3) - Use_TKE_Flag
*      par(11,3) - Add DBZ (DBZ)

*
*      par(01..11,4) - "Radar" segment

*      par(01,4) - Initial radar range (km)
*      par(02,4) - Number of range cells
*      par(03,4) - Transmitted power (watts)
*      par(04,4) - Frequency (GHz)
*      par(05,4) - Pulse width (microsec)
*      par(06,4) - Pulse interval (microsec)
*      par(07,4) - Receiver noise figure (dB)
*      par(08,4) - Receiver losses (dB)
*      par(09,4) - RMS trans. phase jitter (deg)
*      par(10,4) - RMS trans. freq jitter (Hz)
*      par(11,4) - Pulse Sampling Interval (microsec)

*
*      par(01..12,5) - "Antenna and Scan" segment

c      par(01,5) -      Ant Type (1 . 2. or 3)
c      par(02,5) -      Antenna Radius (m)
c      par(03,5) -      Aperture Taper Parameter
c      par(04,5) -      Scan Type (1=Az/El,2=El/Az)
c      par(05,5) -      Scan Rate (deg/sec)
c      par(06,5) -      Az Scan Center (deg)
c      par(07,5) -      El Scan Center (deg)
c      par(08,5) -      No. of Az (El) Scan Lines (odd)
c      par(09,5) -      No. of El (AZ) Bars
c      par(10,5) -      Az Scan Increment (deg)
c      par(11,5) -      El Scan Increment (deg)
c      par(12,5) -      Spare

*
*      par(01..10,6) - "Signal processing" segment

*      par(01,6) - Number of pulses in CPI
*      par(02,6) - Number of A/D bits
*      par(03,6) - AGC gain factor
*      par(04,6) - Processing threshold (dB)
*      par(05,6) - Clutter filter code (-2 to N)
*      par(06,6) - Clutter filter cutoff (m/s)
*      par(07,6) - Number of bins for F-factor averaging
*      par(08,6) - Threshold for F-factor algorithm
*      par(09,6) - AR model order
*      par(10,6) - Spare

*
*      par(01..11,7) - "Data products" segment

*      par(01,7) - Alarm program data code
*      par(02,7) - C.S.V. (plot) output code
*      par(03,7) - IQ data output code
*      par(04,7) - AR coefficient code
*      par(05,7) - AR spectra data code
*      par(06,7) - Fourier spectra data code

```

```
*          par(07,7) - Start scan number
*          par(08,7) - Finish scan number
*          par(09,7) - Start line number
*          par(10,7) - Finish line number
*          par(11,7) - Spare
```

```
* Notes:
```

```
*      1) The input datafile, WXIN40.DAT, consists of seven
* segments as indicated above. Each segment has the same
* structure: 1) a title record identifying the segment but
* containing no data; a blank record; a given number of data
* records; and a final blank record. This convention requires
* the addition of a single blank record to the end of old
* datafiles.
```

```
*      2) Input data are no longer echoed to standard output.
```

```
*-----
```

```
ip_read = 0.0

DO j=1,7
  imax = nrecord(j)
  READ(lun,18,END=110)
  .   dum, dum, (par(i,j),i=1,imax), dum
18   FORMAT(33X,F10.0)
END DO

CLOSE(lun)

ip_read = 1.0

110 CONTINUE

RETURN

ENTRY ip_get(irecord,segment)
```

```
*-----
```

```
* Purpose:
```

```
*      To return a given parameter.
```

```
* Inputs:
```

```
*      irecord - Index of record containing parameter
*      segment - Index of segment containing parameter
```

```
* Outputs:
```

```
*      ip_get - Value of parameter
```


APPENDIX B

SUBROUTINE LISTING

```
MODULE ANT
IMPLICIT NONE
SAVE

C   COMMON/ANT/C1, C2, XK, C12, GAINO
REAL XK, GAINO

END MODULE ANT
```

```
MODULE ANTR
IMPLICIT NONE

SAVE

C      COMMON/ANTR/ANTDAT
      REAL, DIMENSION(1800,2) :: ANTDAT

END MODULE ANTR
```

```
MODULE ATTN
IMPLICIT NONE
SAVE

C      COMMON/ATTN/CON1, CON2, BETA
      REAL CON2, BETA

END MODULE ATTN
```

```
MODULE BFIL
IMPLICIT NONE
SAVE

C      COMMON/BFIL/FC11,FC12,FC21,FC22,FC23
REAL FC11,FC12,FC21,FC22,FC23

END MODULE BFIL
```

```
MODULE CLUT
IMPLICIT NONE
SAVE

C      COMMON/CLUT/RMR, RHOMO, SFR, SFA, IRCODE, CA, IRUSE, JMAXO
C      2      , SIGURB, AREAC, RRM, CO

REAL, DIMENSION (3,3) :: RMR, RRM
REAL, DIMENSION (3) :: RHOMO
REAL, DIMENSION (2) :: CA
REAL, DIMENSION (4) :: CO
REAL SFR, SFA, SIGURB, AREAC
INTEGER, DIMENSION (6) :: IRUSE
INTEGER IRCODE, JMAXO
DATA CO/.331, -4.339E-2, 9.416E-4, -6.180E-6/

END MODULE CLUT
```

```
MODULE CONST
IMPLICIT NONE
SAVE

C      COMMON/CONST/PI, DTR, RTD, CC, FREQ, XLAM
REAL, PARAMETER :: PI=3.141592654
REAL, PARAMETER :: DEGREES_2_RAD=2.*3.141592654/360.
REAL, PARAMETER :: RADIANS_2_DEGR=360./(2.*3.141592654)
REAL, PARAMETER :: CC=2.997925E8          ! speed of light
REAL FREQ, XLAM

END MODULE CONST
```

```
MODULE DISC
IMPLICIT NONE
SAVE

REAL, DIMENSION(80,1000) :: XXD,YYD,RCSD,VXD,VYD,UPD
REAL, DIMENSION(80) :: NND
INTEGER NSEG,IRES
REAL SFRG,SFAZ

END MODULE DISC
```

```
MODULE WIND
IMPLICIT NONE
SAVE

REAL, DIMENSION(101,101) :: UC,WC,ZZ
REAL TIME,DR,DZ
INTEGER IRUN,ISTOP,JSTOP

END MODULE WIND
```

```

MODULE WIND3D

IMPLICIT NONE
SAVE

C   DIMENSIONS FOR WEATHER ARRAYS

INTEGER*4 NMAX
INTEGER*4 NX,NY,NZ,USE_TKE_FLAG

PARAMETER XMAX=512, YMAX=512, ZMAX=128
PARAMETER ARRAY_MAX=XMAX*YMAX*ZMAX
PARAMETER IBDIM=256
PARAMETER ISDIM=8000      !MAXIMUM RAIN SCATTERERS
PARAMETER ICDIM=4000     !MAXIMUM CLUTTER SCATTERERS
PARAMETER IDDIM=200      !MAXIMUM DISCRETE SCATTERERS

C   *** GRID SPACING FOR WEATHER ARRAYS ***

REAL DX,DY,DZP

INTEGER*2 UT (ARRAY_MAX),VT (ARRAY_MAX),WT (ARRAY_MAX),
2          RRF (ARRAY_MAX),VSD_MX (ARRAY_MAX)

REAL ADD_DBZ,XSTART,YSTART,ZSTART

REAL PHASE_ERROR,FREQ_ERROR,CLUTTER_STD_DEV,ELSCAN_RANGE

END MODULE WIND3D

```

```

C
C *****
C *****
C     NAME:      STRNUM()
C
C     PURPOSE:  Generate a uniformly distributed random number
C               between 0 and 1
C
C     INPUT:
C
C     OUTPUT:
C
C *****
C *****
C     SUBROUTINE STRNUM(YY)
C     DATA IFLAG/1/
C     BB=1.
C     P1=YY*317.
C     YY = AMOD(P1,BB)
C     YY=RAND(IFLAG)
C     IFLAG=0
C     RETURN
C     END
C *****
C *****
C     NAME:      ANTPAT()
C
C     PURPOSE:  CALCULATES ANTENNA GAIN AT ANGLE ANG
C
C     INPUT:    ANG IS IN DEGREES - P=1
C               REQUIRES SCI.LIB FOR BESSEL SUBROUTINES
C
C     OUTPUT:   GAIN
C
C *****
C *****
C     SUBROUTINE ANTPAT(A,B,ANG,GAIN)
C     USE CONST
C     USE ANT
C     IMPLICIT DOUBLE PRECISION (A-H,0-Z)
C     COMMON/CONST/PI,DTR,RTD,CC,FREQ,XLAM
C     COMMON/ANT/C1,C2,XK,C12,GAINO
C     U=ABS(XK*A*SIN(ANG*DEGREES_2_RAD))
C     CON=4./(1.+B)
C     GAIN=GAINO
C     IF(U.LT.1.E-12) U=1.E-12
C     CALL BESSN(U,1.,XJ1)
C     CALL BESSN(U,2.,XJ2)
C     XJ1=besj1(U)
C     XJ2=besj2(U)
C     FX=(B*XJ1/U + (1.-B)*2.*XJ2/(U*U))*CON
C     GAIN=FX*FX*GAINO
C     RETURN
C     END
C *****
C *****

```

```

C *****
C     NAME:      NORMAL ()
C
C     PURPOSE:  Provide a random number in accordance with normal
C               distribution with a standard deviation of unity
C
C     INPUT:    none
C
C     OUTPUT:   XNRN - random number
C
C *****
C *****
C     SUBROUTINE NORMAL (XNRN)
C     DATA RN/.237/
C     SUM=0.
C     DO 20 INN=1,12
C     CALL STRNUM (RN)
C     RNZM=RN-.5
20    SUM=SUM+RNZM
C     XNRN=SUM
C     RETURN
C     END
C
C *****
C *****
C     NAME:      NORNG ()
C
C     PURPOSE:  GENERATE A SEQUENCE OF NUMBERS NORMALLY
C               AND RANDOMLY DISTRIBUTED OVER THE INTERVAL -3 TO 3 FROM
C               UNIFORMLY DISTRIBUTED RANDOM NUMBERS BY THE METHOD OF LINEAR
C               APPROXIMATION TO THE INVERSE OF THE ACCUMULATIVE
C               NORMAL DISTRIBUTION FUNCTION.
C
C     INPUT:
C
C     OUTPUT:
C
C *****
C *****
C     SUBROUTINE NORNG (R,P)
C     DIMENSION Y(6), X(6), S(5)
C     DATA Y/0.,.0228,.0668,.1357,.2743,.5/,
C     &X/-3.01,-2.0,-1.5,-1.0,-.6,0./,
C     &S/43.8596,11.3636,7.25689,2.891352,2.65887/
C     CALL STRNUM (R)
C     P = R
C     I = 1
C     IF (P.GT.0.5) P = 1.0-R
2    IF (P.LT.Y(I + 1)) GO TO 8
C     I = I + 1
C     GO TO 2
8    P = ((P-Y(I))*S(I) + X(I))
C     IF (R.GE.0.5) P = -P
C     RETURN
C     END
C
C *****

```

```

C *****
C     NAME:      CROSS ()
C
C     PURPOSE:  TAKES THE CROSS PRODUCT A X B = R
C
C     INPUT:    A, B
C
C     OUTPUT:   R
C
C*****
C*****
C     SUBROUTINE CROSS (A, B, R)
C     IMPLICIT DOUBLE PRECISION (A-H, O-Z)
C     DIMENSION A (3), B (3), R (3)
C     R (1)=A (2) *B (3) -A (3) *B (2)
C     R (2)=A (3) *B (1) -A (1) *B (3)
C     R (3)=A (1) *B (2) -A (2) *B (1)
C     RETURN
C     END
C
C *****
C *****
C     NAME:      MEG ()
C
C     PURPOSE:  CREATES MATRIX FOR CONVERSION FROM EARTH CENTERED TO
C              GEOGRAPHIC COORD.
C
C     INPUT:    PHI
C              OM
C
C     OUTPUT:   R
C
C*****
C*****
C     SUBROUTINE MEG (PHI, OM, R)
C     IMPLICIT DOUBLE PRECISION (A-H, O-Z)
C     DIMENSION R (3, 3)
C     DTR=2.*3.141592654/360.
C     X=PHI*DTR
C     Y=OM*DTR
C     SX=SIN (X)
C     CX=COS (X)
C     SY=SIN (Y)
C     CY=COS (Y)
C     R (1, 1)=-SX*CY
C     R (2, 1)=-SY
C     R (3, 1)=-CX*CY
C     R (1, 2)=-SX*SY
C     R (2, 2)=CY
C     R (3, 2)=-CX*SY
C     R (1, 3)=CX
C     R (2, 3)=0.
C     R (3, 3)=-SX
C     RETURN
C     END
C
C *****

```

```

C *****
C     NAME:      SPHER()
C
C     PURPOSE:  RECTANGULAR TO SPHERICAL CONVERSION OF VECTOR
C
C     INPUT:    A - vector with x,y,z coordinates
C
C     OUTPUT:   R - 1 X 3 matrix with radial distance, clock & cone angles
C
C*****
C*****
C     SUBROUTINE SPHER(A,R)
C       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C       DIMENSION A(3),R(3)
C       X=A(1)
C       Y=A(2)
C       Z=A(3)
C       RTD=360./(2.*3.141592654)
C       AM=SQRT(X*X+Y*Y+Z*Z)
C       CONE=(ACOS(Z/AM))*RTD
C       CLOCK=(ATAN2(Y,X))*RTD
C       IF(CLOCK.LT.0.)CLOCK=360.+CLOCK
C       R(1)=AM
C       R(2)=CLOCK
C       R(3)=CONE
C       RETURN
C       END
C
C *****
C *****
C     NAME:      GMPRD()
C
C     PURPOSE:  MULTIPLIES TWO MATRICES A X B = R
C
C     INPUT:    N=ROWS OF A
C               M=COLUMNS OF A,ROWS OF B.
C               L=COL. OF B
C
C     OUTPUT:   R=result matrix
C
C*****
C*****
C     SUBROUTINE GMPRD(A,B,R,N,M,L)
C       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C       DIMENSION A(*),B(*),R(*)
C       IR=0
C       IK=-M
C       DO 10 K=1,L
C         IK=IK+M
C         DO 10 J=1,N
C           IR=IR+1
C           JI=J-N
C           IB=IK
C           R(IR)=0.
C           DO 10 I=1,M
C             JI=JI+N
C             IB=IB+1

```

```

10      R(IR)=R(IR)+A(JI)*B(IB)
      RETURN
      END
C
C *****
C *****
C      NAME:      GMTRA()
C
C      PURPOSE:  TRANSPOSES MATRIX A TO GIVE R.
C
C      INPUT:    A= MATRIX WITH N=ROWS,M=COLS.
C
C      OUTPUT:   R= transpose of matrix A
C
C *****
C *****
      SUBROUTINE GMTRA(A,R,N,M)
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION A(*),R(*)
      IR=0
      DO 10 I=1,N
      IJ=I-N
      DO 10 J=1,M
      IJ=IJ+N
      IR=IR+1
10      R(IR)=A(IJ)
      RETURN
      END
C
C *****
C *****
C      NAME:      ROT()
C
C      PURPOSE:  CREATES ROTATION MATRIX.
C
C      INPUT:    IROT SPECIFIES AXIS OF ROTATION
C                1=X AXIS, 2=Y AXIS, 3=Z AXIS
C
C      OUTPUT:   R=result matrix
C
C *****
C *****
      SUBROUTINE ROT(IROT,ANG,R)
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION R(3,3)
      X=ANG*(2.*3.141592654)/360.
      C=COS(X)
      S=SIN(X)
      DO 10 I=1,3
      DO 20 J=1,3
      R(I,J)=0.
20      CONTINUE
10      CONTINUE
      IF (IROT .NE. 1) GO TO 200
      R(2,2)=C
      R(3,2)=-S
      R(2,3)=S

```

```

        R(3,3)=C
        R(1,1)=1.
        RETURN
200    IF (IROT .NE. 2) GO TO 300
        R(1,1)=C
        R(3,1)=S
        R(1,3)=-S
        R(3,3)=C
        R(2,2)=1.
        RETURN
300    R(1,1)=C
        R(2,1)=-S
        R(1,2)=S
        R(2,2)=C
        R(3,3)=1.
        RETURN
    END

```

```

C
C *****
C *****
C     NAME:      EMAT()
C
C     PURPOSE:  GENERATES EULER MATRIX.
C
C     INPUT:    P=PHI,T=THETA,PS=PSI
C
C     OUTPUT:   R=Euler matrix
C
C *****
C *****

```

```

    SUBROUTINE EMAT(P,T,PS,R)
    DIMENSION R(3,3)
    DTR=3.141592654/180.
    X=P*DTR
    Y=T*DTR
    Z=PS*DTR
    CP=COS(X)
    SP=SIN(X)
    CT=COS(Y)
    ST=SIN(Y)
    CPS=COS(Z)
    SPS=SIN(Z)
    R(1,1)=CT*CPS
    R(2,1)=-CP*SPS + SP*ST*CPS
    R(3,1)=SP*SPS + CP*ST*CPS
    R(1,2)=CT*SPS
    R(2,2)=CP*CPS + SP*ST*SPS
    R(3,2)=-SP*CPS + CP*ST*SPS
    R(1,3)=-ST
    R(2,3)=SP*CT
    R(3,3)=CP*CT
    RETURN
    END

```

```

C
C *****
C *****
C     NAME:      BESSN()

```

```

C
C   PURPOSE: FIND THE BESSEL FUNCTION OF INTEGER OR
C             FRACTIONAL ORDER FOR ANY X GREATER THAN ZERO.
C
C   INPUT:
C
C   OUTPUT:
C
C*****
C*****
SUBROUTINE BESSN(X1,B1,Y1)
  DIMENSION C(6), D(6), S(26)
  DATA S/-3.375,54.4921875,-272.953125,354.375,-2.0625,13.7578125,
&-19.6875,-1.0625,1.875,-.375,.0030381944,-.4861111111,10.286458
&33333,-58.,78.75,.0005580357,-.4241071428,3.6026785714,
&-5.625,.0125,-.35,.75,.0416666666,.25,3.1415926535,1.57
&07963267/
  Y1 = 0.0
  IF(B1.LT.0.0) RETURN
2  IF(X1.GE.14.9) GO TO 51
  C1 = B1 + 1.0
  CALL GAMMA(C1,Z1)
  TERM1 = (X1*0.5)** B1/Z1
  P1 = 1.0
  Y1 = TERM1
10  TERM1 = -TERM1*X1*X1/(4.0*(B1 + P1)*P1)
  Y1 = Y1 + TERM1
  P1 = P1 + 1.0
  IF(ABS(Y1)*1.E-11.LT.ABS(TERM1)) GO TO 10
  RETURN
C 51  IF(B1.GT.1.0) GO TO 53
51  CONTINUE
  IFLAG = 1
  X = X1
  B = B1
  Y = Y1
  GO TO 1
15  RETURN
53  IB1 = B1
  A1 = B1 - FLOAT(IB1)
  IFLAG = 2
  X = X1
  B = A1
  XX1 = 0.
  Y = XX1
  GO TO 1
16  P1 = A1 - 1.0
  IFLAG = 3
  X = X1
  B = P1
  YY1 = 0
  Y = YY1
  GO TO 1
60  Y1 = 2.0*A1**X1/X1 - YY1
  A1 = A1 + 1.0
  IF(A1.EQ.B1) RETURN
  YY1 = XX1

```

```

XX1 = Y1
GO TO 60
1 CONTINUE
A = B*B - 0.25
T = 1.0/X
T2 = T*T
C(1) = ((S(1)*A+S(2))*A+S(3))*A+S(4))*A
C(2) = ((S(5)*A+S(6))*A+S(7))*A
C(3) = (S(8)*A+S(9))*A
C(4) = S(10)*A
BBO = ((C(1)*T2+C(2))*T2+C(3))*T2+C(4))*T2*T2+1.0
BB = BBO*SQRT(2.*T/(S(25)*SQRT(1.-A*T2)))
D(1) = (((S(11)*A+S(12))*A+S(13))*A+S(14))*A+S(15))*A
D(2) = (((S(16)*A+S(17))*A+S(18))*A+S(19))*A
D(3) = ((S(20)*A+S(21))*A + S(22))*A
D(4) = (S(23)*A + S(24))*A
D(5) = .5*A
AA = (((D(1)*T2 + D(2))*T2+D(3))*T2+D(4))*T2 +D(5))*T2+1.0
AA = X-(B + .5)*S(26)
BB= SQRT(2./(S(25)*X))
Y1 = BB*COS(AA)
GO TO (15,16,60),IFLAG
END
C
C *****
C *****
C NAME: GAMMA()
C
C PURPOSE: FIND THE GAMMA FUNCTION FOR
C ANY VARIABLE X GREATER THAN ZERO.
C
C INPUT: X
C
C OUTPUT: GAM
C
C *****
C *****
SUBROUTINE GAMMA(X,GAM)
DIMENSION B(9)
DATAB/.035868343,-.193527818,.482199394,-.756704078,
&.918206857,-.897056937,.988205891,-.577191652,1.00/
N = 0
GAM1 = X
X1 = X
IF(X.LE.0.0) GO TO 5
IF(X.LE.1.0) GO TO 4
IF(X.LE.2.0) N = 1
1 X1 = X1 - 1.0
IF(X1.LT.2.0) GO TO 2
GAM1 = GAM1*X1
GO TO 1
2 IF(X1.GT.1.0) X1 = X1 - 1.00
3 ND = 8
CALL PLYEVL (B,ND,X1,GAM)
IF(N.EQ.1) RETURN
GAM = GAM*GAM1
RETURN

```

```

4   GAM1 = 1.0/X1
   GO TO 3
5   GAM = 9999999.9
   RETURN
   END

C
C *****
C *****
C   NAME:      GAMMA ()
C
C   PURPOSE:  EVALUATES ANY SINGLE VARIABLE POLYNOMIAL OF THE FORM
C             A (1) *X**ND+A (2) *X** (ND-1)+... +A (ND) *X+A (ND+1)
C
C   INPUT:    A - vector of coefficients
C
C   OUTPUT:   ANS
C
C *****
C *****
C   SUBROUTINE PLYEVL (A,ND,X,ANS)
C   DIMENSION A (1)
C   ANS=A (1)
C   IF (ND.EQ.0) RETURN
C   DO 20 I=1,ND
20  ANS=ANS*X+A (I+1)
   RETURN
   END

C
C *****
C *****
C   NAME:      FFT ()
C
C   PURPOSE:   - COMPUTE THE FAST FOURIER TRANSFORM, GIVEN A
C             COMPLEX VECTOR OF LENGTH EQUAL TO A POWER
C             OF TWO
C   INPUT:     Z      - COMPLEX VECTOR OF LENGTH N=2**M
C             WHICH CONTAINS ON INPUT THE
C             DATA TO BE TRANSFORMED. ON
C             OUTPUT,A CONTAINS THE FOURIER
C             COEFFICIENTS.
C             M      - N = 2**M IS THE NUMBER OF DATA POINTS.
C             M= +N FFT WILL PERFORM FOURIER
C             TRANSFORM.
C             M= -N FFT WILL PERFORM INVERSE
C             TRANSFORM.
C             IWK    - WORK AREA VECTOR OF LENGTH M+1.
C   PRECISION - SINGLE
C   LATEST REVISION - APRIL 16, 1980
C
C   OUTPUT:
C
C *****
C *****
C   SUBROUTINE FFT (Z,M,IWK)
C   DIMENSION IWK (*),Z (*),Z0 (2),Z1 (2),Z2 (2),Z3 (2)
C   COMPLEX Z,ZA0,ZA1,ZA2,ZA3,AK2
C   EQUIVALENCE      (ZA0,Z0 (1)),(ZA1,Z1 (1)),(ZA2,Z2 (1)),(ZA3,Z3 (1))

```

```

*,
*1,Z1(2)), (A0,Z0(1)), (B0,Z0(2)), (A1,Z1(1)), (B
*3,Z3(2)), (A2,Z2(1)), (B2,Z2(2)), (A3,Z3(1)), (B
DATA SQ,SK,CK /.70710678118655,.38268343236509,
* .92387953251129/
DATA TWOPI/6.2831853071796/,ZERO/0.0/,ONE/1.0/
C SQ=SQRT2/2,SK=SIN(PI/8),CK=COS(PI/8)
C TWOPI=2*PI

SIGN = 1.0
IF (M .LT. 0) SIGN = -1.0
M= IABS(M)
MP = M+1
N = 2**M
IF (SIGN .EQ. 1.0) GO TO 4
DO 2 I=1,N
    Z(I) = CONJG(Z(I))
2 CONTINUE
4 IWK(1)=1
MM = (M/2)*2
KN = N+1
C INITIALIZE WORK VECTOR

DO 5 I=2,MP
    IWK(I) = IWK(I-1)+IWK(I-1)
5 CONTINUE
RAD=TWOPI/N
MK = M - 4
KB = 1
IF (MM .EQ. M) GO TO 15
K2 = KN
K0 = IWK(MM+1) + KB
10 K2 = K2 - 1
K0 = K0 - 1
AK2 = Z(K2)
Z(K2) = Z(K0) - AK2
Z(K0) = Z(K0) + AK2
IF (K0 .GT. KB) GO TO 10
15 C1 = ONE
S1 = ZERO
JJ = 0
K = MM - 1
J = 4
IF (K .GE. 1) GO TO 30
GO TO 9005
20 IF (IWK(J) .GT. JJ) GO TO 25
JJ = JJ - IWK(J)
J = J-1
IF (IWK(J) .GT. JJ) GO TO 25
JJ = JJ - IWK(J)
J = J - 1
K = K + 2
GO TO 20
25 JJ = IWK(J) + JJ
J = 4
30 ISP = IWK(K)
IF (JJ .EQ. 0) GO TO 40
C RESET TRIGONOMETRIC PARAMETERS

C2 = JJ * ISP * RAD

```

```

C1 = COS(C2)
S1 = SIN(C2)
35 C2 = C1 * C1 - S1 * S1
    S2 = C1 * (S1 + S1)
    C3 = C2 * C1 - S2 * S1
    S3 = C2 * S1 + S2 * C1
40 JSP = ISP + KB
C
C
                                DETERMINE FOURIER COEFFICIENTS
                                IN GROUPS OF 4
DO 50 I=1,ISP
    K0 = JSP - I
    K1 = K0 + ISP
    K2 = K1 + ISP
    K3 = K2 + ISP
    ZA0 = Z(K0)
    ZA1 = Z(K1)
    ZA2 = Z(K2)
    ZA3 = Z(K3)
    IF (S1 .EQ. ZERO) GO TO 45
    TEMP = A1
    A1 = A1 * C1 - B1 * S1
    B1 = TEMP * S1 + B1 * C1
    TEMP = A2
    A2 = A2 * C2 - B2 * S2
    B2 = TEMP * S2 + B2 * C2
    TEMP = A3
    A3 = A3 * C3 - B3 * S3
    B3 = TEMP * S3 + B3 * C3
45  TEMP = A0 + A2
    A2 = A0 - A2
    A0 = TEMP
    TEMP = A1 + A3
    A3 = A1 - A3
    A1 = TEMP
    TEMP = B0 + B2
    B2 = B0 - B2
    B0 = TEMP
    TEMP = B1 + B3
    B3 = B1 - B3
    B1 = TEMP
    Z(K0) = CMPLX(A0+A1,B0+B1)
    Z(K1) = CMPLX(A0-A1,B0-B1)
    Z(K2) = CMPLX(A2-B3,B2+A3)
    Z(K3) = CMPLX(A2+B3,B2-A3)
50 CONTINUE
    IF (K .LE. 1) GO TO 55
    K = K - 2
    GO TO 30
55 KB = K3 + ISP
C
C
                                CHECK FOR COMPLETION OF FINAL
                                ITERATION
    IF (KN .LE. KB) GO TO 9005
    IF (J .NE. 1) GO TO 60
    K = 3
    J = MK
    GO TO 20
60 J = J - 1

```

```

C2 = C1
IF (J .NE. 2) GO TO 65
C1 = C1 * CK + S1 * SK
S1 = S1 * CK - C2 * SK
GO TO 35
65 C1 = (C1 - S1) * SQ
S1 = (C2 + S1) * SQ
GO TO 35
9005 CONTINUE
IF (SIGN .EQ. 1.0) GO TO 75
XN = N
DO 70 I=1,N
Z(I)=CONJG(Z(I))/XN
70 CONTINUE
75 CALL FFRDR2(Z,M,IWK)
IF(SIGN.EQ.-1.0) M = -M
RETURN
END

```

```

C
C *****
C *****
C     NAME:      FFRDR2 ()
C
C     PURPOSE:  - THIS SUBROUTINE PERMUTES A COMPLEX DATA VECTOR
C                IN REVERSE BINARY ORDER TO NORMAL ORDER. THE
C                ROUTINE CAN ALSO BE USED TO PERMUTE A COM-
C                PLEX DATA VECTOR IN NORMAL ORDER TO REVERSE
C                BINARY ORDER SINCE THE PERMUTATION IS SYM-
C                METRIC.
C
C     INPUT:    Z COMPLEX VECTOR OF LENGTH N=2**M WHICH CONTAINS ON INPUT
C                THE DATA TO BE PERMUTED.
C                M          - N=2**M IS THE NUMBER OF DATA POINTS.
C                IWK        - WORK AREA VECTOR OF LENGTH M+1
C
C     OUTPUT:   Z CONTAINS THE PERMUTED DATA VECTOR.
C
C     LATEST REVISION      - MARCH 16, 1973
C *****
C *****
C     SUBROUTINE FFRDR2 (Z,M,IWK)
C     DIMENSION IWK(*),Z(*)
C     COMPLEX Z,TEMP
C
C     IF(M .LE. 1) GO TO 9005
C     MP = M+1
C     JJ = 1
C
C                                     INITIALIZE WORK VECTOR
C
C     IWK(1) = 1
C     DO 5 I = 2,MP
C         IWK(I) = IWK(I-1) * 2
C 5 CONTINUE
C     N4 = IWK(MP-2)
C     IF (M .GT. 2) N8 = IWK(MP-3)
C     N2 = IWK(MP-1)
C     LM = N2

```

```

      NN = IWK(MP)+1
      MP = MP-4
C
      J = 2
      DETERMINE INDICES AND SWITCH A*S
10  JK = JJ + N2
      TEMP= Z(J)
      Z(J)=Z(JK)
      Z(JK)= TEMP
      J = J+1
      IF (JJ .GT. N4) GO TO 15
      JJ = JJ + N4
      GO TO 35
15  JJ = JJ - N4
      IF (JJ .GT. N8) GO TO 20
      JJ = JJ + N8
      GO TO 35
20  JJ = JJ - N8
      K = MP
25  IF (IWK(K) .GE. JJ) GO TO 30
      JJ = JJ - IWK(K)
      K = K - 1
      GO TO 25
30  JJ = IWK(K) + JJ
35  IF (JJ .LE. J) GO TO 40
      K = NN - J
      JK = NN - JJ
      TEMP= Z(J)
      Z(J) = Z(JJ)
      Z(JJ) = TEMP
      TEMP = Z(K)
      Z(K) = Z(JK)
      Z(JK)= TEMP
40  J = J + 1
C
      CYCLE REPEATED UNTIL LIMITING NUMBER
C
      OF CHANGES IS ACHIEVED
      IF (J .LE. LM) GO TO 10
9005 RETURN
C** THIS PROGRAM VALID ON FTN4 AND FTN5 **
      END
C
C *****
C *****
C      NAME:      MATINV()
C
C      PURPOSE:  FIND THE INVERSE AND/OR SOLVES SIMULTANEOUS EQUATIONS,
C                OR NEITHER, AND CALCULATES A DETERMINANT OF A REAL MATRIX.
C
C      INPUT:
C
C      OUTPUT:
C
C *****
C *****
      SUBROUTINE MATINV (ISOL, IDSOL, NR, NC, A, MRA, KWA, DET)
      DIMENSION A(1), KWA(1)
      IR = NR
      ISOL = 1

```

```

IDSOL = 1
IF(NR.LE.0) GO TO 330
IF((IR-MRA).GT.0) GO TO 330
IC = IABS(NC)
IF ((IC - IR).LT.0) IC = IR
IBMP = 1
JBMP = MRA
KBMP = JBMP + IBMP
NES = IR*JBMP
NET = IC*JBMP
IF(NC) 10,330,20
10 MDIV = JBMP + 1
   IRIC = IR - IC
   GO TO 30
20 MDIV = 1
30 MAD = MDIV
   MSER = 1
   KSER = IR
   MZ = 1
   DET = 1.0
40 PIV = 0.
   I = MSER
50 IF (( I - KSER).GT.0) GO TO 70
   IF((ABS(A(I))-PIV).LE.0.) GO TO 60
   PIV = ABS(A(I))
   IP = I
60 I = I + IBMP
   GO TO 50
70 IF(PIV.EQ.0.) GO TO 340
   IF(NC.LT.0) GO TO 80
   I = IP-((IP - 1)/JBMP)*JBMP
   J = MSER - ((MSER - 1)/JBMP)*JBMP
   JJ = MSER/KBMP + 1
   II = JJ + (IP -MSER)
   KWA(JJ) = II
   GO TO 90
80 I = IP
   J = MSER
90 IF (IP - MSER) 330,120,100
100 IF ((J - NET).GT.0) GO TO 110
   PSTO = A(I)
   A(I) = A(J)
   A(J) = PSTO
   I = I + JBMP
   J = J + JBMP
   GO TO 100
110 DET = - DET
120 PSTO = A(MSER)
   DET = DET*PSTO
130 IF (DET.eq.0.) GOTO 150
140 PSTO = 1./PSTO
   GO TO 160
150 IDSOL = 1
   ISOL = 2
   RETURN
160 CONTINUE
   A(MSER) = 1.0

```

```

I = MDIV
170 IF((I - NET).GT.0) GO TO 180
A(I) = A(I)*PSTO
I = I + JBMP
GO TO 170
180 IF((MZ - KSER).GT.0) GO TO 210
IF((MZ-MSER).EQ.0) GO TO 200
I = MAD
J = MDIV
PSTO = A(MZ)
IF(PSTO.EQ.0.) GO TO 200
A(MZ) = 0.
190 IF((J-NET).GT.0) GO TO 200
A(I) = A(I) - A(J)*PSTO
J = J + JBMP
I = I + JBMP
GO TO 190
200 MAD = MAD + IBMP
MZ = MZ + IBMP
GO TO 180
210 continue
C 210 NEED A TEST HERE.....CALL OVERFL(IVF)
C GO TO (350,220),IVF
CCCCCCC NEED AT TEST HERE, ANYHOW
220 KSER = KSER + JBMP
IF ((KSER-NES).GT.0) GO TO 260
MSER = MSER + KBMP
IF(NC.LT.0) GO TO 230
MDIV = MDIV + IBMP
MZ = ((MSER - 1)/JBMP)*JBMP + 1
MAD = 1
GO TO 40
230 MDIV = MDIV + KBMP
IF(IRIC.NE.0) GO TO 240
MZ = MSER + IBMP
GO TO 250
240 MZ = ((MSER - 1)/JBMP)*JBMP + 1
250 MAD = MZ + JBMP
GO TO 40
260 IF(NC.LT.0) RETURN
JR = IR
270 IF(JR) 330,360,280
280 IF(KWA(JR) - JR) 330,320,290
290 K = (JR - 1)*JBMP
J = K + IR
L = (KWA(JR) - 1)*JBMP + IR
300 IF(J - K) 330,320,310
310 PSTO = A(L)
A(L) = A(J)
A(J) = PSTO
J = J - IBMP
L = L - IBMP
GO TO 300
320 JR = JR - 1
GO TO 270
330 ISOL = 3
RETURN

```



```

C *****
C *****
C     NAME:      ANG360 (ANG)
C
C     PURPOSE:  Converts angle to range 0-360
C
C     INPUT:    ang = angle in degrees +/- 180
C
C     OUTPUT:   ang in degrees 0-360
C*****
C*****

```

```

SUBROUTINE ANG360 (ANG)
REAL ANG

IF (ANG.LT.0.) ANG = ANG + 360.

RETURN
END

```

```

C *****
C *****
C     NAME:      FSIZE2 ()
C
C     PURPOSE:  Determine number of bytes to write to a file
C
C     INPUT:    FNAME, ICODE, NBPL, NS, NL, NB, NP, ISSCAN, IFSCAN, ISLINE, IFLINE
C
C     OUTPUT:   NUM_SCANS_IN_OUTPUT, NUM_LINES_IN_OUTPUT
C*****
C*****

```

```

SUBROUTINE FSIZE2 (FNAME, ICODE, NBPL, NS, NL, NB, NP, ISSCAN, IFSCAN,
2 ISLINE, IFLINE, NUM_SCANS_IN_OUTPUT, NUM_LINES_IN_OUTPUT)
IMPLICIT INTEGER (I-N)
CHARACTER(50) FNAME
CHARACTER(10) LB
LB=' KIBYTES'
FUDGEF=1.5
IF (ICODE .EQ. 0) RETURN
KS=NS
KL=NL
KB=NB
KP=NP
IDLINES=IFLINE-ISLINE + 1
IDSCANS=IFSCAN-ISSCAN + 1
IF (ICODE.EQ.2.OR.ICODE.EQ.4) KL=IDLINES
IF (ICODE.EQ.3.OR.ICODE.EQ.4) KS=IDSCANS
NUM_SCANS_IN_OUTPUT=KS
NUM_LINES_IN_OUTPUT=KL
KSL=KS*KL
NBYTES=FUDGEF*KSL*KB*KP*NBPL/1000
WRITE (*, *) FNAME, NBYTES, LB
RETURN
END

```

C

```

C *****
C *****
C     NAME:      WRTFLG ()
C
C     PURPOSE: Determine if data should be written to an output file and
C               if so, set IFLG=1
C
C     INPUT:   SCAN LIMITS AND CODE (ICODE) FOR FILE TYPE
C
C     OUTPUT:  IFLG
C
C *****
C *****
C     SUBROUTINE WRTFLG (ISSCAN, IFSCAN, ISLINE, IFLINE, ITM, JS, ICODE, IFLG)
C     IFLG=0
C     IF (ICODE.EQ.0) RETURN
C     IF (ICODE.EQ.1) THEN
C         IFLG=1
C         RETURN
C     ENDIF
C     IF (ICODE.EQ.4.AND.ITM.LE.IFSCAN.AND.ITM.GE.ISSCAN
C &       .AND.JS.LE.IFLINE.AND.JS.GE.ISLINE) IFLG=1
C     IF (ICODE.EQ.2.AND.JS.LE.IFLINE.AND.JS.GE.ISLINE) IFLG=1
C     IF (ICODE.EQ.3.AND.ITM.LE.IFSCAN.AND.ITM.GE.ISSCAN) IFLG=1
C     RETURN
C     END
C
C *****
C *****
C     NAME:      RDDISC2 ()
C
C     PURPOSE:  VERSION 5.1 - READS DISCRETE TARGET FILE
C               MODIFIED FOR VERSION 5.1 TO USE DATA COORDINATES
C
C     INPUT:    TARGET FILE
C
C     OUTPUT:   POSITION, VELOCITY AND REFLECTIVITY OF TARGETS
C
C *****
C *****
C     SUBROUTINE RDDISC2
C     USE DISC
C     USE CLUT
C     IMPLICIT INTEGER (I-N)
C     DIMENSION R1 (3), R2 (3)
C     READ (17, *) NSEG
C     READ (17, *) (NND (I), I=1, NSEG)
C     READ (17, *) ((XXD (I, J), J=1, NND (I)), I=1, NSEG)
C     READ (17, *) ((YYD (I, J), J=1, NND (I)), I=1, NSEG)
C     READ (17, *) ((UPD (I, J), J=1, NND (I)), I=1, NSEG)
C     READ (17, *) ((RCSD (I, J), J=1, NND (I)), I=1, NSEG)
C     READ (17, *) ((VXD (I, J), J=1, NND (I)), I=1, NSEG)
C     READ (17, *) ((VYD (I, J), J=1, NND (I)), I=1, NSEG)
C     WRITE (*, *) 'DISCRETE TARGET FILE READ'
C
C     RETURN

```

```

        END
C
C *****
C *****
C     NAME:      P()
C
C     PURPOSE:  Write a 3 x 3 matrix to the screen
C
C     INPUT:    XMAT - matrix to write
C
C     OUTPUT:   none
C
C *****
C *****
      SUBROUTINE P(XMAT)
      DIMENSION XMAT(3,3)
      WRITE(*,*) ' '
      WRITE(*,12) XMAT(1,1), XMAT(1,2), XMAT(1,3)
      WRITE(*,12) XMAT(2,1), XMAT(2,2), XMAT(2,3)
      WRITE(*,12) XMAT(3,1), XMAT(3,2), XMAT(3,3)
12  FORMAT(1X,3F8.2)
      RETURN
      END
C *****
C *****
C     NAME:      SETVAR()
C
C     PURPOSE:  Copy variables to output array which will be written
C              to a file later.
C
C     INPUT:    V1 - V24 are variables to write
C              MM - index into output array
C              OV - output array
C
C     OUTPUT:   OV - modified by the addition of new data
C
C *****
C *****
      SUBROUTINE SETVAR(V1,V2,V3,V4,V5,V6,V7,V8,V9,V10,V11,V12,V13,
2      V14,V15,V16,V17,V18,V19,V20,V21,V22,V23,V24,MM,OV)
      IMPLICIT INTEGER(I-N)
      DIMENSION OV(24,128)
      OV(1,MM)=V1
      OV(2,MM)=V2
      OV(3,MM)=V3
      OV(4,MM)=V4
      OV(5,MM)=V5
      OV(6,MM)=V6
      OV(7,MM)=V7
      OV(8,MM)=V8
      OV(9,MM)=V9
      OV(10,MM)=V10
      OV(11,MM)=V11
      OV(12,MM)=V12
      OV(13,MM)=V13
      OV(14,MM)=V14

```

```

OV (15,MM)=V15
OV (16,MM)=V16
OV (17,MM)=V17
OV (18,MM)=V18
OV (19,MM)=V19
OV (20,MM)=V20
OV (21,MM)=V21
OV (22,MM)=V22
OV (23,MM)=V23
OV (24,MM)=V24
RETURN
END
C
C *****
C *****
C     NAME:      FFAC ()
C
C     PURPOSE:  CALCULATES F-FACTOR OR HAZARD INDEX
C
C     INPUT:    FC
C              VSAVE
C              VPP
C              VSP
C              VTRU
C
C     OUTPUT:   FPP
C              FSP
C              FTRU
C
C *****
C *****
C     SUBROUTINE FFAC (FC, VSAVE, VPP, VSP, VTRU, FPP, FSP, FTRU)
C     IMPLICIT INTEGER (I-N)
C     DIMENSION VSAVE (3, 2)
C     FPP=-FC* (VPP-VSAVE (1, 1))
C     FSP=-FC* (VSP-VSAVE (2, 1))
C     FTRU=-FC* (VTRU-VSAVE (3, 1))
C     RETURN
C     END
C
C *****
C *****
C     NAME:      AVRHAZ ()
C
C     PURPOSE:  Makes use of the OV array to average the hazard factor
C              over several range bins
C
C     INPUT:    OV - output variable array
C              NBINS - number of range bins
C
C     OUTPUT:
C
C *****
C *****
C     SUBROUTINE AVRHAZ (OV, NN, NBINS)
C     IMPLICIT INTEGER (I-N)
C     DIMENSION OV (24, 128), FPP (128), FSP (128), FTR (128), FTO (128)

```

```

IF(NN .LE. 2)RETURN
JSTART=(NN+1)/2
JSTOP=NBINS-JSTART+1
DO 20 J=1,NBINS
IF (J.GE.JSTART.AND.J.LE.JSTOP) THEN
  IDEX2=J+(NN-1)/2
  IDEX1=J-(NN-1)/2
  FPP(J)=0.
  FSP(J)=0.
  FTR(J)=0.
  FTO(J)=0.
  DO 55 K=IDEX1,IDEX2
  FPP(J)=OV(15,K)/NN+FPP(J)
  FSP(J)=OV(16,K)/NN+FSP(J)
  FTR(J)=OV(18,K)/NN+FTR(J)
55  FTO(J)=OV(20,K)/NN+FTO(J)
ELSE
  FPP(J)=0.
  FSP(J)=0.
  FTR(J)=0.
  FTO(J)=0.
ENDIF
20  CONTINUE
DO 30 J=1,NBINS
OV(15,J)=FPP(J)
OV(16,J)=FSP(J)
OV(18,J)=FTR(J)
OV(20,J)=FTO(J)
30  CONTINUE
RETURN
END

```

```

C *****
C *****
C   NAME:      HAZLES ()
C
C   PURPOSE:  COMPUTES HAZARD FACTOR BASED ON WEIGHTED LEAST-SQUARES
C             ALGORITHM.  IV IS THE INDEX OF THE VELOCITY MEASUREMENTS IN
C             ARRAY OV AND IO IS THE INDEX OF THE DESIRED OUTPUT HAZARD
C             CALCULATION.  AS CURRENTLY DIMENSIONS, FIVE VELOCITY
MEASUREMENTS
C             ARE USED TO COMPUTE THE HAZARD.
C             THE RESIDUALS ARE COMPUTED AS "RES" AND PROVISION IS MADE
C             FOR AN INPUT THRESHOLD, "THRES".
C
C   INPUT:    OV -
C             IV -
C             IO -
C             NBINS -
C             BINSZ -
C             CON -
C             RES -
C             THRES -
C
C   OUTPUT:
C
C *****

```

```

C*****
SUBROUTINE HAZLES (OV, IV, IO, NBINS, BINSZ, CON, RES, THRES)
IMPLICIT INTEGER (I-N)
DIMENSION OV (24, 128), FPP (128)
DIMENSION V (5), WT (5, 5), TRES (128)
NN=5
JSTART=(NN+1)/2
JSTOP=NBINS-JSTART+1
DO 20 J=1, NBINS
IF (J.GE.JSTART.AND.J.LE.JSTOP) THEN
  IDEX2=J+(NN-1)/2
  IDEX1=J-(NN-1)/2
  FPP (J)=0.
  LL=0
  DO 55 K=IDEX1, IDEX2
  LL=1+LL
  V (LL)=OV (IV, K)
  SIGMA=OV (14, K)
  IF (SIGMA .LT. .01) SIGMA=.01
  WT (LL, LL)=1./ (SIGMA**2)
55  CONTINUE
  CALL HAZALG (WT, V, SLOPE, RES, BINSZ)
  FPP (J)=SLOPE*CON
  TRES (J)=RES
  IF (RES.GT.THRES) FPP (J)=0.
ELSE
  TRES (J)=0.
  FPP (J)=0.
ENDIF
20  CONTINUE
DO 30 J=1, NBINS
OV (IO, J)=FPP (J)
IF (TRES (J).GT. 999.) TRES (J)=999.
OV (21, J)=TRES (J)
30  CONTINUE
RETURN
END

C
C *****
C *****
C      NAME:      HAZALG ()
C
C      PURPOSE:  THIS ROUTINE FINDS THE SLOPE AND RESIDUALS
C                OF THE MEASUREMENT MATRIX V USING THE WEIGHT
C                MATRIX WT FOR FIVE MEASUREMENTS.
C
C      INPUT:    WT -
C                V -
C                SLOPE -
C                RES -
C                BINSZ -
C
C      OUTPUT:
C
C *****
C *****
SUBROUTINE HAZALG (WT, V, SLOPE, RES, BINSZ)

```

```

      IMPLICIT INTEGER (I-N)
      DIMENSION WT (5,5),V (5),B (2),A (5,2),AT (2,5)
      DIMENSION T1 (5,2),T2 (2,2),KWA (5)
      DIMENSION R1 (5),R2 (5),R3 (5)
      NR=5
      NC=5
      MRA=5
      DO I=1,5
      DO J=1,5
      IF (I.NE.J) WT (I,J)=0.
      END DO
      END DO
      DO I=1,5
      A (I,1)=1.
      A (I,2)=(I-1)*BINSZ
      END DO
C
C *****CALCULATE SLOPE*****
C
C      CALL MATINV (ISOL,IDSOL,NR,NC,WT,MRA,KWA,DET)
      CALL GMPRD (WT,A,T1,5,5,2)
      CALL GMTRA (A,AT,5,2)
      CALL GMPRD (AT,T1,T2,2,5,2)
      CALL MATINV (ISOL1,IDSOL1,2,2,T2,2,KWA,DET1)
      CALL GMPRD (WT,V,R1,5,5,1)
      CALL GMPRD (AT,R1,R2,2,5,1)
      CALL GMPRD (T2,R2,B,2,2,1)
      SLOPE=B (2)
C
C *****FIND RESIDUALS*****
C
      CALL GMPRD (A,B,R1,5,2,1)
      DO I=1,5
      R2 (I)=V (I)-R1 (I)
      END DO
      CALL GMPRD (WT,R2,R3,5,5,1)
      CALL GMPRD (R2,R3,R1,1,5,1)
      RES=R1 (1)/5.
      RETURN
      END
C
C *****
C *****
C      NAME:      DFILTER ()
C
C      PURPOSE:  THIS SUBROUTINE IS A HIGH-PASS DIGITAL FILTER FOR
C               THE I & Q SIGNALS USED TO SUPPRESS CLUTTER SIGNALS
C
C      INPUT:    U1 -
C               U2 -
C               F -
C               NT -
C
C      OUTPUT:
C
C *****
C *****

```

```

SUBROUTINE DFILTER(U1,U2,F,NT)
IMPLICIT INTEGER (I-N)
DIMENSION U1(512),U2(512),FU1(512),FU2(512)
C1=1.-F
FU1(1)=U1(1)
FU2(1)=U2(1)
DO 785 I=2,NT
FU1(I)=C1*FU1(I-1) + U1(I) - U1(I-1)
FU2(I)=C1*FU2(I-1) + U2(I) - U2(I-1)
785 CONTINUE
DO 786 I=1,NT
U1(I)=FU1(I)
U2(I)=FU2(I)
786 CONTINUE

RETURN
END

C
C *****
C *****
C NAME: EFILTER()
C
C PURPOSE: THIS SUBROUTINE IS A 1ST OR 2ND ORDER BUTTERWORTH FILTER FOR
C THE I & Q SIGNALS USED TO SUPPRESS CLUTTER SIGNALS
C
C INPUT: U1 -
C U2 -
C IFIL -
C NT -
C
C OUTPUT:
C
C *****
C *****
SUBROUTINE EFILTER(U1,U2,IFIL,NT)
USE BFIL
IMPLICIT INTEGER (I-N)
DIMENSION U1(512),U2(512),FU1(512),FU2(512)
C COMMON/BFIL/FC11,FC12,FC21,FC22,FC23
FU1(1)=U1(1)
FU2(1)=U2(1)
FU1(2)=U1(2)
FU2(2)=U2(2)
ILO=-2
IHI=-1
IF(IFIL.LT.ILO.OR.IFIL.GT.IHI)THEN
WRITE(*,*)'RETURN FROM EFILTER - NO FILTERING DONE'
RETURN
ELSEIF(IFIL.EQ.-2)THEN
DO 785 I=3,NT
FU1(I)=FC21*FU1(I-1)+FC22*FU1(I-2)+FC23*
2 (U1(I)-2.*U1(I-1)+U1(I-2))
FU2(I)=FC21*FU2(I-1)+FC22*FU2(I-2)+FC23*
2 (U2(I)-2.*U2(I-1)+U2(I-2))
785 CONTINUE
WRITE(*,*)'TWO POLE BUTTERWORTH FILTER COMPLETED'
ELSEIF(IFIL.EQ.-1)THEN

```

```

DO 789 I=2,NT
FU1 (I)=FC11*FU1 (I-1)+FC12*(U1 (I)-U1 (I-1))
FU2 (I)=FC11*FU2 (I-1)+FC12*(U2 (I)-U2 (I-1))
789 CONTINUE
WRITE (*,*) 'SINGLE POLE BUTTERWORTH FILTER COMPLETED'
ENDIF
DO 786 I=1,NT
U1 (I)=FU1 (I)
U2 (I)=FU2 (I)
786 CONTINUE
RETURN
END

C
C *****
C *****
C NAME: FILCO ()
C
C PURPOSE: CALCULATES COEFFICIENTS FOR 1ST AND 2ND
C ORDER DIGITAL BUTTERWORTH FILTERS
C
C INPUT: VMAX -
C VCO -
C
C OUTPUT:
C
C *****
C *****
SUBROUTINE FILCO (VMAX,VCO)
USE BFIL
IMPLICIT INTEGER (I-N)
C COMMON/BFIL/FC11,FC12,FC21,FC22,FC23
DATA B1,C1,B2,C2,D2/.293408,.353296,.677496,.253921,.144106/
PI=3.14159
R=VCO/VMAX
ALP= -COS (.5*(1.+PI*R))/COS (.5*(1.-PI*R))
FC11=-(ALP+B1)/(1.+ALP*B1)
FC12=C1*(1.-ALP)/(1.+ALP*B1)
DEM=1. + ALP*B2 + ALP*ALP*C2
FC21=-(2.*ALP*(1.+C2) + B2*(1.+ALP*ALP))/DEM
FC22=-(ALP*ALP + ALP*B2 + C2)/DEM
FC23=(D2*(1.-ALP)**2)/DEM
c WRITE (*,4) FC11,FC12,FC21,FC22,FC23
4 FORMAT (1X,5F15.4)
RETURN
END

C
C *****
C *****
C NAME: ADCONV ()
C
C PURPOSE: THIS SUBROUTINE CALCULATES THE
C GAIN REQUIRED TO SET THE SIGNALS
C TO A +/- UNITY LEVEL AND THEN QUANTIZES THE SIGNALS
C IN ACCORDANCE WITH THE NUMBER OF LEVELS OF THE A/D.
C GFAC IS THE LEVEL DESIRED FOR AVERAGE SIGNAL
C THIS ROUTINE ASSUMES AGC IN EACH RANGE BIN
C

```

```

C      INPUT:   U1 -
C              U2 -
C              NP -
C              NSAMP -
C              NLEV -
C              GFAC -
C              SDNSE -
C              GAIN -
C
C      OUTPUT:
C
C*****
C*****
      SUBROUTINE ADCONV(U1,U2, NP, NSAMP, NLEV, GFAC, SDNSE, GAIN)
      IMPLICIT INTEGER (I-N)
      DIMENSION U1(512), U2(512)
C
C *****CALCULATE AVERAGE POWER & REQUIRED GAIN***
C
C      WRITE (*, *) NP, NSAMP, NLEV, GFAC, SDNSE
      GAINM=GFAC/SDNSE
      PAV=0.
      DO 5 I=1, NSAMP
5      PAV=PAV+(U1(I)*U1(I) + U2(I)*U2(I))
      GAIN=GFAC/SQRT(PAV)
C      WRITE (*, *) GAIN, GAINM, PAV
      IF (GAIN .GE. GAINM) GAIN=GAINM
      GAIN1=GAIN*NLEV
C
C*****QUANTIZE SIGNALS AND RENORMALIZE TO ORIGINAL LEVELS*****
C
      DO 10 I=1, NP
      T1=U1(I)
      T2=U2(I)
      XNUM=FLOAT(INT(GAIN1*U1(I)))
      U1(I)=XNUM/GAIN1
      XNUM=FLOAT(INT(GAIN1*U2(I)))
      U2(I)=XNUM/GAIN1
C      WRITE (*, *) T1, U1(I), T2, U2(I)
10     CONTINUE
      RETURN
      END
C
C*****
C*****
      NAME:      ANTFLAT()
C
C      PURPOSE:  CALCULATES GAIN OF FLAT (COLLINS TYPE) ANTENNA
C                BEAMWIDTH IS 3.5 DEGREES
C
C      INPUT:    A & B ARE NOT USED
C                ANG IS IN DEGREES
C                GAIN IN DB
C
C      OUTPUT:
C
C*****

```

```

C*****
SUBROUTINE ANTFLAT (ANG, GAIN)
IMPLICIT INTEGER (I-N)
DATA A1, A2, A3, A4, A5, A6, A7, A8, A9/1.75, 7., 12.25, 14.,
2 15.75, 28., 40.25, 42., 43.75/
GO=34.5
G1=9.5
AA=ABS (ANG)
GAIN=0.
IF (AA.LT.A1) THEN
  GAIN=GO-5.*ALOG10 (EXP (5.55* (AA/3.5) **2) )
ELSEIF (AA.GE.A1.AND.AA.LT.A2) THEN
  GAIN=GO+11.3-8.175*AA
ELSEIF (AA.GE.A2.AND.AA.LT.A3) THEN
  GAIN=G1+11.3-8.175* (A4-AA)
ELSEIF (AA.GE.A3.AND.AA.LT.A5) THEN
  GAIN=G1-5.* ALOG10 (EXP (5.55* ( (A4-AA) /3.5) **2) )
ELSEIF (AA.GE.A5.AND.AA.LT.A6) THEN
  GAIN=G1+11.3-8.175* (AA-A4)
ELSEIF (AA.GE.A6.AND.AA.LT.A7) THEN
  GAIN=G1+11.3-8.175* (A8-AA)
ELSEIF (AA.GE.A7.AND.AA.LT.A9) THEN
  GAIN=G1-5.*ALOG10 (EXP (5.55* ( (A8-AA) /3.5) **2) )
ELSEIF (AA.GE.A9) THEN
  GAIN=G1+11.3-8.175* (AA-A8)
55  ENDIF
IF (GAIN .LT. 0.) GAIN=0.
GAIN=10.** (GAIN/10.)
RETURN
END

C
C *****
C *****
C NAME:    HORIZ ()
C
C PURPOSE: CALCULATES ANTENNA POINTING ANGLE WITH RESPECT
C           TO HORIZON IN DEGREES
C
C INPUT:
C
C OUTPUT:
C
C*****
C*****
SUBROUTINE HORIZ (RFIN, H2, HO)
IMPLICIT INTEGER (I-N)
DIMENSION RFIN (3, 3) , R1 (3, 3) , H2 (3) , H1 (3) , HO (3)
H1 (1) =1.
H1 (2) =0.
H1 (3) =0.
CALL GMTRA (RFIN, R1, 3, 3)
CALL GMPRD (R1, H1, H2, 3, 3, 1)
C WRITE (*, *) H2 (1) , H2 (2) , H2 (3)
CALL SPHER (H2, HO)
C WRITE (*, *) HO (1) , HO (2) , HO (3)
RETURN
END

```

```

C
C *****
C *****
C     NAME:      WXANAL ( )
C
C     PURPOSE:  CALCULATES VELOCITY USING SPECTRAL AVERAGING
C
C     INPUT:
C
C     OUTPUT:
C
C *****
C *****
C     SUBROUTINE WXANAL (U1, U2, N, DT, STHRES, IMAGDB, VSP)
C     USE CONST
C     IMPLICIT INTEGER (I-N)
C     DIMENSION U1 (512), U2 (512), IWK (10)
C     DIMENSION RE (512), XIMAG (512), XMAG (512), IMAGDB (512)
C     DIMENSION X (512)
C     COMPLEX Z (512)
C     COMMON/CONST/PI, DTR, RTD, CC, FREQ, XLAM
C     XKTTMS=.5148
C
C     DO 10 I=1, N
10    Z (I) = CMPLX (U1 (I), U2 (I))
C     CONTINUE
C
C     DELF=1. / (N*DT)
C     DELV=DELF*CC / (2.*FREQ)
C     XN=N
C     M=(ALOG10 (XN) /ALOG10 (2.)) + .5
C     CALL FFT (Z, M, IWK)
C
C     DO 20 I=1, N
C     RE (I) = REAL (Z (I))
C     XIMAG (I) = AIMAG (Z (I))
C     XMAG (I) = (RE (I) *RE (I) + XIMAG (I) *XIMAG (I)) / (XN*XN)
C     XMAG (I) = MAX (XMAG (I), 1.E-18)
C     IMAGDB (I) = INT (10.*ALOG10 (XMAG (I)))
20    CONTINUE
C
C     S=0.
C     DO 889 I=1, N
C     IF (I.LE.N/2) II=I+N/2
C     IF (I.GT.N/2) II=I-N/2
C     X (II) = XMAG (N-I+1)
889    CONTINUE
C
C     SUM1=0.
C     SUM2=0.
C     DO 200 I=2, N
C     SUM1=SUM1 + X (I)
C     VELI=(I-1-N/2) *DELV
C     SUM2=SUM2 + VELI*X (I)
200    CONTINUE
C     S=SUM1*N / (N-1)
C     VNN2=(N/2)

```

```

VSP=SUM2/SUM1
IF(S .LT. STHRES)VSP=0.
C WRITE (*,*)VSP,S,STHRES
RETURN
END

C
C *****
C *****
C NAME: PPP()
C
C PURPOSE: CALCULATES VELOCITY & WIDTH USING PULSE-PAIR PROCESSING
C
C INPUT: U1,U2,N,PRR,VEL,STHRES
C
C OUTPUT: VPP,WPP
C *****
C *****
SUBROUTINE PPP(U1,U2,N,PRR,STHRES,VPP,WPP)
USE CONST
IMPLICIT INTEGER (I-N)
DIMENSION U1(512),U2(512)
C COMMON/CONST/PI,DTR,RTD,CC,FREQ,XLAM
SUMT=1.E-50
XKTTMS=.5148
PI4=4.*PI
CONS=XLAM/(2.*PI*PRR*1.414)
SUMR=0.
SUMI=0.
SUMSQ=0.
M=N-1
VPP=0.
WPP=0.
DO 10 I=1,M
SUMR=SUMR + U1(I)*U1(I+1) + U2(I)*U2(I+1)
SUMI=SUMI + U1(I)*U2(I+1) - U1(I+1)*U2(I)
SUMSQ=SUMSQ + U1(I)*U1(I) + U2(I)*U2(I)
10 CONTINUE
IF(SUMSQ .LT. SUMT)SUMSQ=SUMT
C IF(SUMR .LT. SUMT)SUMR=SUMT
C IF(SUMI .LT. SUMT)SUMI=SUMT
VPP=(XLAM/(PI4*PRR))*ATAN2(SUMI,SUMR)
R1=(1./M)*SQRT(SUMR*SUMR + SUMI*SUMI)
S=(1./M)*SUMSQ
IF(S .LT. STHRES)VPP=0.
WPP=0.
IF(R1 .EQ. 0.)GO TO 45
TEST=ALOG(S/R1)
SGN=-1.
IF(TEST .GT. 0.)SGN=1.
WPP=SGN*CONS*SQRT((ABS(TEST)))
C IF(WPP .GT. 5.)VPP=0.
IF(S .LT. STHRES)WPP=0.
C WRITE (*,*)VPP,WPP,R1,S,STHRES
45 RETURN
END
C

```

```

C *****
C *****
C     NAME:      ATTEN ()
C
C     PURPOSE:  CALCULATES 2-WAY PATH LOSS DUE TO MOISTURE
C
C     INPUT:    RH,ZO,BINSZ,BX,BY,ANGM, XLOSS
C
C     OUTPUT:
C
C *****
C *****
      SUBROUTINE ATTEN(RH,ZO,BINSZ,BX,BY,ANGM, XLOSS)
      USE ATTN
      USE CONST
      IMPLICIT INTEGER (I-N)
      DIMENSION RH(3)
C     COMMON/ATTN/CON1,CON2,BETA
      DUM=0.
C     TMAX=16.E6
      XLOSS=1.
      X=RH(1)
      Y=RH(2)
      Z=RH(3)
      XMAG=SQRT(X*X+Y*Y+Z*Z)
      SUM=0.
      SO=XMAG
      DO 10 I=1,50
          S=SO-BINSZ*(2.*I - 1.)
C         WRITE(*,6)SO,S
          FORMAT(1X,2F10.0)
          IF(S .LE. 1000.)GO TO 35
          CON=S/XMAG
          XX=X*CON      !in AC coordinates
          YY=Y*CON
          ZZ=Z*CON-ZO
          IF(ZZ .LE. 0.)GO TO 10
C         TEST=(XX-BX)*(XX-BX) + (YY-BY)*(YY-BY)
C         IF(TEST .GE. TMAX)GO TO 10

          AXX=XX+BX
          AYY=YY+BY
          AZZ=ZZ
          CALL GET WEATHER PARAMETERS
              .      (U,V,W,DBZ,DELTA WIND,AXX,AYY,AZZ,3,ANGM)
          DBZZ=DBZ      ! FOR DEBUGGING
          IF(DBZ .LT. 1.)RETURN      !CLB 12/5/99
          ZZZ=10**(DBZ/10.)
          RR=(ZZZ/200.)**.625
          XK=CON2*(RR**BETA)
          SUM=SUM - XK*BINSZ*2.  !USES 2*BINSZ FOR INTEGRATION INCREMENT
C         WRITE(*,40)I,XX,YY,ZZ,S,DBZ,RR,XK,SUM
      10  CONTINUE
      40  FORMAT(1X,I3,4F7.0,2F5.1,2E10.2)
C *****2-WAY LOSS FACTOR*****
      35  XLOSS=EXP(2.*SUM)
C     ATTNDB=10.*ALOG10(XLOSS)      !FOR CHECKING VALUES

```

```

C      WRITE (*, *) XLOSS, ATTNDDB
      RETURN
      END

C
C *****
C *****
C      NAME:      VELTRU ()
C
C      PURPOSE:  CALCULATES VELOCITY COMPONENT ALONG PROJECTED
C                ANTENNA BEAM CENTER
C
C      INPUT:    RFIN, VX, VELIN, ROO, RRG, ZO, BX, BY, FVER, VTRU, VVER, ANGM
C
C      OUTPUT:
C
C *****
C *****
      SUBROUTINE VELTRU (RFIN, VX, VELIN, ROO, RRG, ZO, BX, BY, FVER, VTRU, VVER
&      , ANGM)
      IMPLICIT INTEGER (I-N)
      DIMENSION RFIN (3, 3), VX (3), RHOBM2 (3), R1 (3, 3), RHOANT (3)
      XKTTMS=.5148
      DO 10 I=1, 3
10    RHOANT (I)=0.
      VTRU=0.
      RHOANT (1)=RRG*1000.
      CALL GMTRA (RFIN, R1, 3, 3)
      CALL GMPRD (R1, RHOANT, RHOBM2, 3, 3, 1)
      XX=RHOBM2 (1)
      YY=RHOBM2 (2)
      ZZZ=RHOBM2 (3)
      RHOMAG=SQRT (XX*XX + YY*YY + ZZZ*ZZZ)
      ZZ=ZZZ-ZO
      IF (ZZ .LE. 0.) go to 20

      AXX=XX + BX
      AYY=YY + BY
      AZZ=ZZ
      CALL GET_WEATHER_PARAMETERS
      .      (U, V, W, DBZ, DELTA_WIND, AXX, AYY, AZZ, 2, ANGM)

      VXX=VX (1) -U
      VYY=VX (2) -V
      VZZ=VX (3) -W
      VVER=W
      FVER=0.
      IF (VELIN.GT.50.) FVER=-W/ (VELIN*XKTTMS)      !CLB 12/5/99
      VTRU= (VXX*XX+VYY*YY+VZZ*ZZZ) /RHOMAG) -VELIN

c      WRITE (*, *) 'xx, yy, zzz, AXX, AYY, AZZ '
c      WRITE (*, 35) xx, yy, zzz, axx, ayy, AZZ

      35  FORMAT (1X, 6F7.0)
      20  RETURN
      END

C *****

```

```

C *****
C   NAME:      VEL_WIND ()
C
C   PURPOSE:  CALCULATES AVERAGE VELOCITY COMPONENT ALONG
C             FLIGHT PATH OF AIRCRAFT
C
C   INPUT:    RFIN, VX, VELIN, ROO, RRG, ZO, BX, BY, FVER, VTRU, VVER, ANGM
C
C   OUTPUT:
C
C *****

```

```

SUBROUTINE VEL_WIND(XO, YO, ZO, XVEL, YVEL, AVX, AVY, VEL)
IMPLICIT INTEGER (I-N)
DIMENSION VTEMP(10), VXTEMP(10), VYTEMP(10)
KKTMS=.5148

```

```

VXX=XVEL      !COMPONENTS OF AC GND VELOCITY
VYY=YVEL
DELS=100.
VVV=SQRT(VXX**2 + VYY**2)
NUMDELS=6

```

```

DO II=1, NUMDELS
  SS=100. + (II-1)*DELS
  XAC=SS*VXX/VVV      !AC COORDINATES
  YAC=SS*VYY/VVV
  AXX=XAC+XO          !WEATHER COORDINATES
  AYY=YAC+YO
  AZZ=-ZO
  XMAG=SQRT(XAC**2 + YAC**2)
  CALL GET_WEATHER_PARAMETERS
2      (U, V, W, DBZ, DELTA_WIND, AXX, AYY, AZZ, 2, ANGM)
  VTEMP(II) = ((U*XAC + V*YAC) / XMAG)      !WIND VELOCITY
  VXTEMP(II) = U
  VYTEMP(II) = V
END DO

```

```

SUM=0.
SUMX=0.
SUMY=0.
DO II=1, NUMDELS      !GET AVERAGE WIND OVER PATH
  SUM=SUM + VTEMP(II)
  SUMX=SUMX + VXTEMP(II)
  SUMY=SUMY + VYTEMP(II)
END DO
VEL=SUM/NUMDELS
AVX=SUMX/NUMDELS
AVY=SUMY/NUMDELS

```

```

20  RETURN
    END

```

```

C *****
C *****
C   NAME:      GETBW ()
C
C *****

```

```

C      PURPOSE: GETS 3 DB BEAMWIDTH OF ANTENNAS IN DEGREES
C
C      INPUT:
C
C      OUTPUT:
C
C*****
C*****
      SUBROUTINE GETBW (ITYPE, A, B, BW)
      GO=0.
      BW=0.
      DO 10 I=1,75
      ANG=(I-1)*.1
      IF (ITYPE .EQ. 1) CALL ANTPAT (A, B, ANG, GAIN)
      IF (ITYPE .EQ. 2) CALL ANTFLAT (ANG, GAIN)
      IF (ITYPE .EQ. 3) CALL ANTREAL (ANG, 90., GAIN)
      GDB=10.*ALOG10 (GAIN)
      IF (I .EQ. 1) GO=GDB
      TEST=GO-GDB
      IF (TEST .GT. 3.) GO TO 20
10     CONTINUE
20     BW=ANG*2.
      RETURN
      END
C
C *****
C *****
      NAME:      READANT ()
C
C      PURPOSE: READS ANTENNA PATTERN DATA FROM UNIT 16
C
C      INPUT:
C
C      OUTPUT:
C
C*****
C*****
      SUBROUTINE READANT
      USE ANTR
      IMPLICIT INTEGER (I-N)
C      DIMENSION ANTDAT (1800,2)
C      COMMON/ANTR/ANTDAT
      II=0
      DO 10 I=1,1800
      II=I
      READ (16,*,ERR=999) DUM,T1,T2
      ANTDAT (II,1)=T1
      ANTDAT (II,2)=T2
C      WRITE (*,*) II, DUM, ANTDAT (II,1), ANTDAT (II,2)
C      CALL PAUSE
10     CONTINUE
20     FORMAT (F10.2, F10.2, F10.3/)
      WRITE (*,*) 'ANTENNA DATA READ'
      RETURN
999   WRITE (*,*) 'ERROR IN READING ANT FILE, II= ', II
C      WRITE (*,*) DUM, T1, T2
C      CALL PAUSE

```

```
RETURN
END
```

```
C
C *****
C *****
C     NAME:      ANTREAL ()
C
C     PURPOSE:  CALCULATES ANTENNA GAIN FROM DATA FILE
C
C     INPUT:
C
C     OUTPUT:
C
C *****
C *****
C     SUBROUTINE ANTREAL (ANG, THT, GAIN)
C     USE CONST
C     USE ANTR
C     IMPLICIT INTEGER (I-N)
C     DIMENSION ANTDAT (1800, 2)
C     COMMON/CONST/PI, DTR, RTD, CC, FREQ, XLAM
C     COMMON/ANTR/ANTDAT
C     GAINO=34.6
C     THTR=THT*DEGREES_2_RAD
C     IANG=INT (10.*ANG)
C     E=COS (THTR)
C     H=SIN (THTR)
C     IPHIE=IANG+900
C     IPHIH=IPHIE
C     IF (E.LE.0.) IPHIE=900-IANG
C     IF (H.LE.0.) IPHIH=900-IANG
C     IE=1801-IPHIE
C     IH=1801-IPHIH
C
C     IF (IE.LE.1) IE=1          !9/25/01 CLB TO PREVENT RUN ERROR
C     IF (IH.LE.1) IH=1
C
C     IF (IE.GT.1800) IE=1800   !9/25/01 CLB TO PREVENT RUN ERROR
C     IF (IH.GT.1800) IH=1800
C
C     write (*, *) ang, tht, ie, ih
C     WE=E*E
C     WH=H*H
C     GE=GAINO+ANTDAT (IE, 1)
C     GH=GAINO+ANTDAT (IH, 2)
C     PE=10.** (GE/10.)
C     PH=10.** (GH/10.)
C     GAIN=PE*WE + PH*WH
C     RETURN
C     END
```

```
C *****
C *****
C     NAME:      READ_DATABASES ()
C
```

```

c     PURPOSE: Read and store weather databases
C
C     INPUT:  Four ( or five) database files
C
C     OUTPUT: Four ( or five) arrays of dimension NX x NY x NZ filled
C             with data.
C
C *****
C *****
      SUBROUTINE READ_DATABASES
      USE WIND3D
      IMPLICIT INTEGER*4 (I-N)
      CHARACTER(80) buf,buf1,buf2

      WRITE(*,*) 'READING DATABASES'

c *** READ AND DISCARD 3 HEADER LINES IN EACH FILE ***
      read(9,*)  buf,nx,buf1,ny,buf2,nz  ! U velocity file
      read(9,*)  buf,dx,buf1,dy,buf2,dzp
      read(9,*)  buf,XSTART,buf1,YSTART,buf2,ZSTART

      read(19,*) buf           ! V velocity file
      read(19,*) buf
      read(19,*) buf

      read(10,*) buf          ! W velocity file
      read(10,*) buf
      read(10,*) buf

      read(11,*) buf         ! Reflectivity file
      read(11,*) buf
      read(11,*) buf

      NMAX=NX*NY*NZ

      read(9,*)  ( UT(I),  I=1,NMAX)
      read(19,*) ( VT(I),  I=1,NMAX)
      read(10,*) ( WT(I),  I=1,NMAX)
      read(11,*) ( RRF(I), I=1,NMAX)

700  format(A80)
713  format(A8,e12.4)           !zstart
715  format(6(I4,1x))         !data

C*** FIX TO CORRECT BAD DATA IN NCAR FILE ***
      DO I=1,NX
         IF(RRF(I) .GT.800.) RRF(I)=RRF(I-1)
         IF(RRF(I) .LT.-1000.) RRF(I)=RRF(I-1)
      END DO

C*** READ TKE DATABASE IF IT WILL BE USED ****
      IF(USE_TKE_FLAG .EQ. 1) THEN
         read(30,*) buf
         read(30,*) buf
         read(30,*) buf
         read(30,*) ( VSD_MX(I), I=1,NMAX)
      ENDIF

```

```

WRITE(*,*) 'DATABASES READ SUCCESSFULLY'

RETURN
END

C *****
C *****
C NAME:     NORMAL2()
C
C PURPOSE:  Provide a random number in accordance with normal
C           distribution with a standard deviation of unity
C
C INPUT:    none
C
C OUTPUT:   XNRN - random number
C
C *****
C *****
C SUBROUTINE NORMAL2 (XNRN)
C DATA RN/.459/
C SUMM=0.
C DO INN=1,12
C   CALL STRNUM(RN)
C   RNZM=RN-.5
C   SUMM=SUMM+RNZM
C END DO
C XNRN=SUMM
C RETURN
C END
C
C *****
C *****
C NAME:     GET_WEATHER_PARAMETERS()
C
C PURPOSE:  CALCULATE THE THREE CARTESIAN WIND COMPONENTS OF A TIME-
C           FROZEN NON-SYMETRIC 3-D WEATHER MODEL BY SPECIFYING THE LOCATION
C           OF AN AIRPLANE RELATIVE TO THE WEATHER SYSTEM'S COORDINATES.
C           THIS SYSTEM IS ASSUMED TO BE A RIGHT-HANDED ONE WITH Z-AXIS
C           POINTING AWAY FROM THE GROUND. OTHERWISE, THE CALCULATED WIND
C           COMPONENTS NEED TO BE MODIFIED ACCORDINGLY.
C INPUT:    (1) AX, AY, AND AZ (IN METERS) ARE THE COORDINATES FOR THE
C           LOCATION OF THE DATA POINT (SCATTERER).
C           (2) N IS A COUNTER THAT WHEN
C               N=2 THE WIND COMPONENTS INCLUDING DELTA_WIND ARE
CALCULATED.
C               N=3 THE REFLECTIVITY IS CALCULATED AND APPEARS AS ZDBZ
C           (3) DELTA_WIND IS THE STD DEV OF ONE WIND COMPONENT DERIVED FROM
C               (1/3)TKE = SQRT (1/6(U**2 + V**2 + W**2))
C
C OUTPUT:   U,V, AND W (IN METERS PER SECOND) ARE THE TWO HORIZONTAL
C           AND THE VERTICAL (DEFINED POSITIVE UPWARD) WIND COM-
C           PONENTS IN THE WEATHER SYSTEM'S COORDINATE SYSTEM.
C           ZDBZ IS THE REFLECTIVITY AT THAT POINT. DELTA_WIND REPRESENTS
C           A MEASURE OF TOTAL KINETIC ENERGY AT THE SAME POINT

```

```

C
C *****
C *****
  SUBROUTINE GET_WEATHER_PARAMETERS
2      (U,V,W,ZDBZ,DELTA_WIND,AX,AY,AZ,N,ANGM)
  USE WIND3D
  IMPLICIT INTEGER*4 (I-N)
  INTEGER*4 temp
  DIMENSION RH02(3),RH03(3),R8(3,3),R9(3,3)

      ANGM=0.

C *** rotate windfield by angle ANGM ***
c      CALL ROT(3,ANGM,R8)      !NO ROTATIONS ANTICIPATED IN THIS VERSION

C *** Get coordinates of position in new (rotated about origin) windfield ***
c      RH02(1) = AX
c      RH02(2) = AY
c      RH02(3) = AZ

      RH03(1) = AX
      RH03(2) = AY
      RH03(3) = AZ

c      CALL GMPRD(R8,RH02,RH03,3,3,1)

C *** Find cell containing position. Set index to upper point. ***

      I=0
      J=0
      K=0

      XMAP =RH03(1)-XSTART
      YMAP =RH03(2)-YSTART
      ZMAP =RH03(3)-ZSTART

      IF (XMAP .GE. 0) I = INT(XMAP / DX) + 1 ! array starts at 1, NOT 0
      IF((I .LT. 1) .OR. ((I+1) .GT. NX)) GOTO 11

      IF (YMAP .GE. 0) J = INT(YMAP / DY) + 1
      IF((J .LT. 1) .OR. ((J+1) .GT. NY)) GOTO 11

      IF (ZMAP .GE. 0) K = INT(ZMAP / DZP) + 1
      IF((K .LT. 1) .OR. ((K+1) .GT. NZ)) GOTO 11

C *** Compute relative, normalized position in grid cell

210  x = 1 + (XMAP/DX) - FLOAT(I)
      y = 1 + (YMAP/DY) - FLOAT(J)
      z = 1 + (ZMAP/DZP) - FLOAT(K)

C*****BOUNDS ON X,Y,Z*****
C
      IF(X.GT.1)      X=1
      IF(Y.GT.1)      Y=1

```

```

        IF(Z.GT.1)      Z=1
        IF(X.LT.0.)    X=0.
        IF(Y.LT.0.)    Y=0.
        IF(Z.LT.0.)    Z=0.

        IF (N .NE. 3) GOTO 300

C *****
C                CALCULATE REFLECTIVITY (N=3)
C *****

C Divide by 10 to compensate for Integer*2 variables!!!

        temp = map_index(I,J,K)
        f000 = 10.0 ** ((RRF(temp) / 10.0) / 10.0)

        temp = map_index(I+1,J, K)
        f100 = 10.0 ** ((RRF(temp) / 10.0) / 10.0)

        temp = map_index(I, J+1,K)
        f010 = 10.0 ** ((RRF(temp) / 10.0) / 10.0)

        temp = map_index(I+1,J+1,K)
        f110 = 10.0 ** ((RRF(temp) / 10.0) / 10.0)

        temp = map_index(I, J, K+1)
        f001 = 10.0 ** ((RRF(temp) / 10.0) / 10.0)

        temp = map_index(I+1,J, K+1)
        f101 = 10.0 ** ((RRF(temp) / 10.0) / 10.0)

        temp = map_index(I, J+1,K+1)
        f011 = 10.0 ** ((RRF(temp) / 10.0) / 10.0)

        temp = map_index(I+1,J+1,K+1)
        f111 = 10.0 ** ((RRF(temp) / 10.0) / 10.0)

        f00 = f100*(x-0.0)+f000*(1.0-x)
        f10 = f110*(x-0.0)+f010*(1.0-x)
        f01 = f101*(x-0.0)+f001*(1.0-x)
        f11 = f111*(x-0.0)+f011*(1.0-x)

        f0 = f10*(y-0.0)+f00*(1.0-y)
        f1 = f11*(y-0.0)+f01*(1.0-y)

        ZDBZ = f1*(z-0.0)+f0*(1.0-z)

        IF (ZDBZ .LT. 1.E-5) ZDBZ = 1.E-5
        ZDBZ = 10.0 * ALOG10(ZDBZ) + ADD_DBZ

C      IF(ADD_DBZ.EQ.29.9) ZDBZ = ADD_DBZ      !CLB - CONSTANT DBZ FOR TESTS

C      ZDBZ=20.      !FOR DEBUGGINH

        U = 0.0
        V = 0.0
        W = 0.0

```

GOTO 400

```
C *****
C          CALCULATE WIND COMPONENTS AND DELTA_WIND VALUE (N=2)
C *****
```

```
C *** CALCULATE KINETIC ENERGY ***
```

```
300  IF (USE_TKE_FLAG .EQ.1) THEN
      temp = map_index(I, J, K)
      f000 = VSD_MX(temp)

      temp = map_index(I+1,J, K)
      f100 = VSD_MX(temp)

      temp = map_index(I, J+1,K)
      f010 = VSD_MX(temp)

      temp = map_index(I+1,J+1,K)
      f110 = VSD_MX(temp)

      temp = map_index(I, J, K+1)
      f001 = VSD_MX(temp)

      temp = map_index(I+1,J, K+1)
      f101 = VSD_MX(temp)

      temp = map_index(I, J+1,K+1)
      f011 = VSD_MX(temp)

      temp = map_index(I+1,J+1,K+1)
      f111 = VSD_MX(temp)

      f00 = f100*(x-0.0)+f000*(1.0-x)
      f10 = f110*(x-0.0)+f010*(1.0-x)
      f01 = f101*(x-0.0)+f001*(1.0-x)
      f11 = f111*(x-0.0)+f011*(1.0-x)

      f0 = f10*(y-0.0)+f00*(1.0-y)
      f1 = f11*(y-0.0)+f01*(1.0-y)
```

```
C Divide by 10 to compensate for Integer*2 variables!!!
```

```
      DELTA_WIND = ( f1*(z-0.0)+f0*(1.0-z) ) / 10.0
      ELSE
      DELTA_WIND = 0.0
      ENDIF
```

```
C *** CALCULATE INDIVIDUAL WIND VECTORS ***
```

```
      temp = map_index(I, J, K)
      f000 = UT(temp)

      temp = map_index(I+1,J, K)
      f100 = UT(temp)
```

```

temp = map_index(I, J+1,K)
f010 = UT(temp)

temp = map_index(I+1,J+1,K)
f110 = UT(temp)

temp = map_index(I, J, K+1)
f001 = UT(temp)

temp = map_index(I+1,J, K+1)
f101 = UT(temp)

temp = map_index(I, J+1,K+1)
f011 = UT(temp)

temp = map_index(I+1,J+1,K+1)
f111 = UT(temp)

f00 = f100*(x-0.0)+f000*(1.0-x)
f10 = f110*(x-0.0)+f010*(1.0-x)
f01 = f101*(x-0.0)+f001*(1.0-x)
f11 = f111*(x-0.0)+f011*(1.0-x)

f0 = f10*(y-0.0)+f00*(1.0-y)
f1 = f11*(y-0.0)+f01*(1.0-y)

U = f1*(z-0.0)+f0*(1.0-z)

```

```

C *** CALCULATE V ***
temp = map_index(I, J, K)
f000 = VT(temp)

temp = map_index(I+1,J, K)
f100 = VT(temp)

temp = map_index(I, J+1,K)
f010 = VT(temp)

temp = map_index(I+1,J+1,K)
f110 = VT(temp)

temp = map_index(I, J, K+1)
f001 = VT(temp)

temp = map_index(I+1,J, K+1)
f101 = VT(temp)

temp = map_index(I, J+1,K+1)
f011 = VT(temp)

temp = map_index(I+1,J+1,K+1)
f111 = VT(temp)

f00 = f100*(x-0.0)+f000*(1.0-x)
f10 = f110*(x-0.0)+f010*(1.0-x)
f01 = f101*(x-0.0)+f001*(1.0-x)

```

```

f11 = f111*(x-0.0)+f011*(1.0-x)

f0 = f10*(y-0.0)+f00*(1.0-y)
f1 = f11*(y-0.0)+f01*(1.0-y)

V = f1*(z-0.0)+f0*(1.0-z)

C *** CALCULATE W ***
temp = map_index(I, J, K)
f000 = WT(temp)

temp = map_index(I+1,J, K)
f100 = WT(temp)

temp = map_index(I, J+1,K)
f010 = WT(temp)

temp = map_index(I+1,J+1,K)
f110 = WT(temp)

temp = map_index(I, J, K+1)
f001 = WT(temp)

temp = map_index(I+1,J, K+1)
f101 = WT(temp)

temp = map_index(I, J+1,K+1)
f011 = WT(temp)

temp = map_index(I+1,J+1,K+1)
f111 = WT(temp)

f00 = f100*(x-0.0)+f000*(1.0-x)
f10 = f110*(x-0.0)+f010*(1.0-x)
f01 = f101*(x-0.0)+f001*(1.0-x)
f11 = f111*(x-0.0)+f011*(1.0-x)

f0 = f10*(y-0.0)+f00*(1.0-y)
f1 = f11*(y-0.0)+f01*(1.0-y)

W = f1*(z-0.0)+f0*(1.0-z)

C Divide by 10 to compensate for Integer*2 variables!!!

U=U/10.0
V=V/10.0
W=W/10.0

C V=-10. !FOR DEBUGGING

C *** CONVERT TO WEATHER SYSTEM COORDINATES ****

C RH02(1) = U
C RH02(2) = V
C RH02(3) = W
C CALL GMTRA(R8,R9,3,3) ! Transpose matrix
C CALL GMPRD(R9,RH02,RH03,3,3,1) ! convert to original coordinate

```

```
C      U = RH03(1)
C      V = RH03(2)
```

```
GOTO 400
```

```
C *****
C      SET COMPONENTS TO DEFAULT VALUE IF OUTSIDE OF GRID
C *****
```

```
11  U = 0.0
     V = 0.0
     W = 0.0
```

```
     IF (N .NE. 3) THEN
         ZDBZ = 0.0
     ELSE
         ZDBZ = -50.0
     ENDIF
```

```
400  CONTINUE
     RETURN
     END
```

```
C *****
C *****
```

```
C
C      NAME:      MAP_INDEX()
C
C      PURPOSE:   MAP A VIRTUAL 3-DIMENSIONAL ARRAY INTO A 1-DIMENSIONAL
C                 ARRAY WHICH IS ACTUALLY DECLARED AND USED.  THIS ALLOWS
C                 PROGRAM TO USE AN ARBITRARY SIZE 3-D ARRAY WITHOUT HAVING
C                 TO RECOMPILE THE CODE EACH TIME THE ARRAY DIMENSIONS
C                 CHANGE
C
C      INPUT:     3-D INDEX (I,J,K)
C
C      OUTPUT:    1-D INDEX
```

```
C *****
C *****
```

```
FUNCTION MAP_INDEX(I,J,K)
USE WIND3D
INTEGER*4 I,J,K

MAP_INDEX = I + NX*(J-1) + NX*NY*(K-1)

RETURN
END FUNCTION
```

APPENDIX C

VERIFICATION & VALIDATION

In 1999 a study was performed to verify the proper operation of ADWRS 4.0. See reference 6 for details of the study. Every effort has been undertaken to ensure that the changes made to ADWRS as part of this upgrade have in no way impacted the accuracy or validity of the results. To that end, a data set was processed using both ADWRS 4.0 and ADWRS 5.0 and the results were compared. Visual inspection of the plots reveals no significant differences. Figures 1 and 2 show plots of velocity produced by the "old" and "new" ADWRS. Similarly, figures 3 and 4 show the spectral width, and figures 5 and 6 show the reflectivity.

In addition, data provided by TASS was processed and the output was compared with the original data. Figures 7 and 8 show the results for reflectivity.

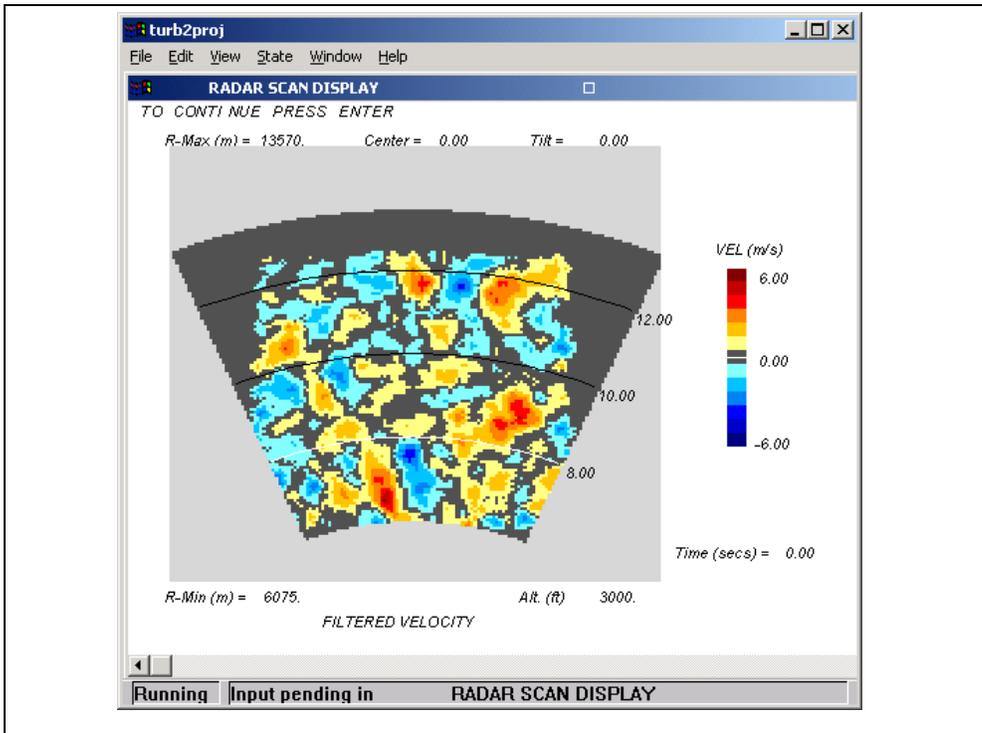


Figure 1 -Velocity Plot Produced by ADWRS 4.0

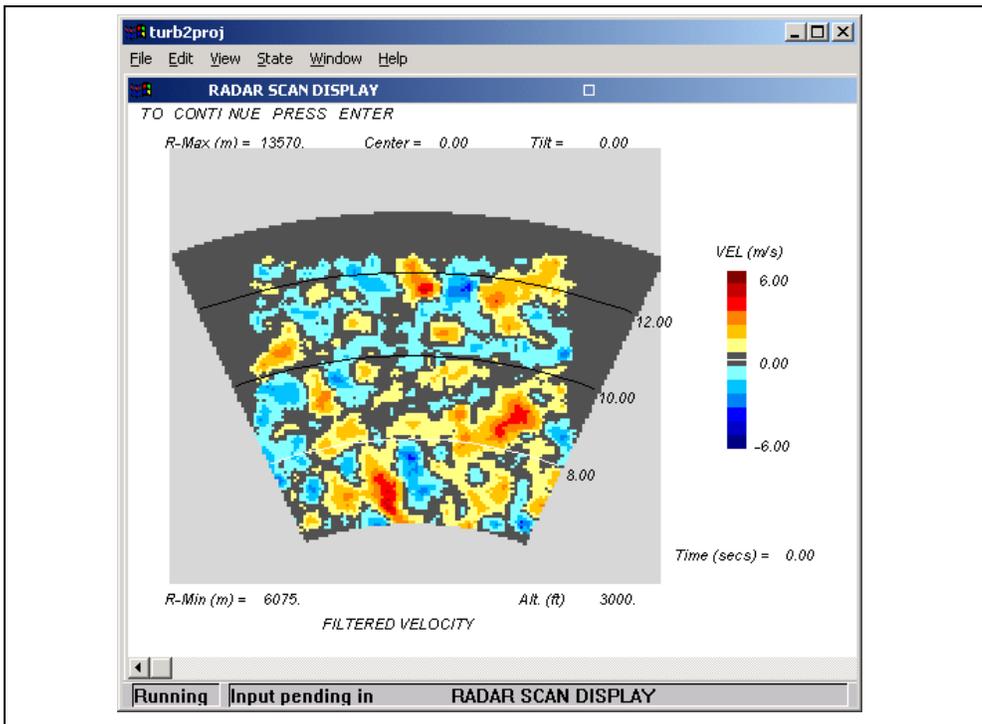


Figure 2 -Velocity Plot Produced by ADWRS 5.0

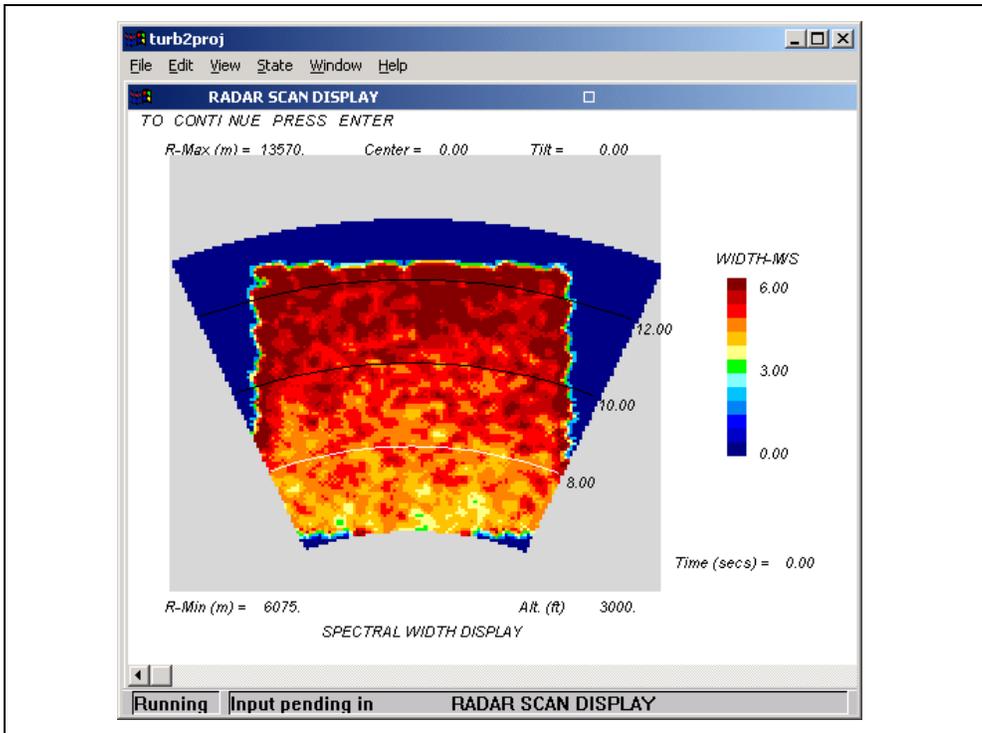


Figure 3 - Spectral Width Produced by ADWRS 4.0

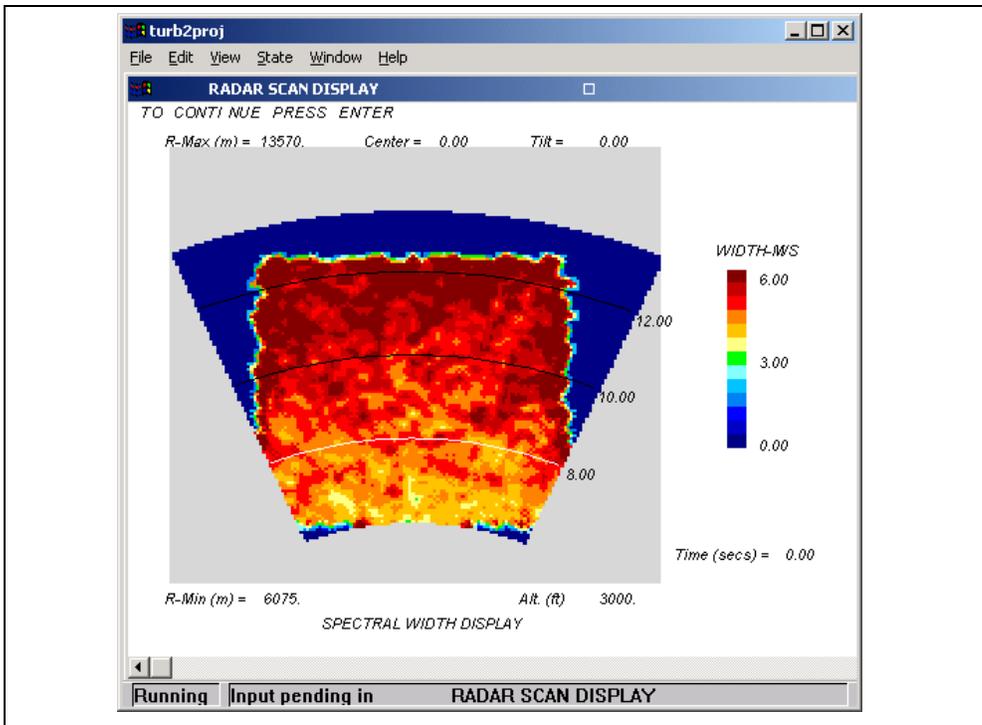


Figure 4 - Spectral Width Produced by ADWRS 5.0

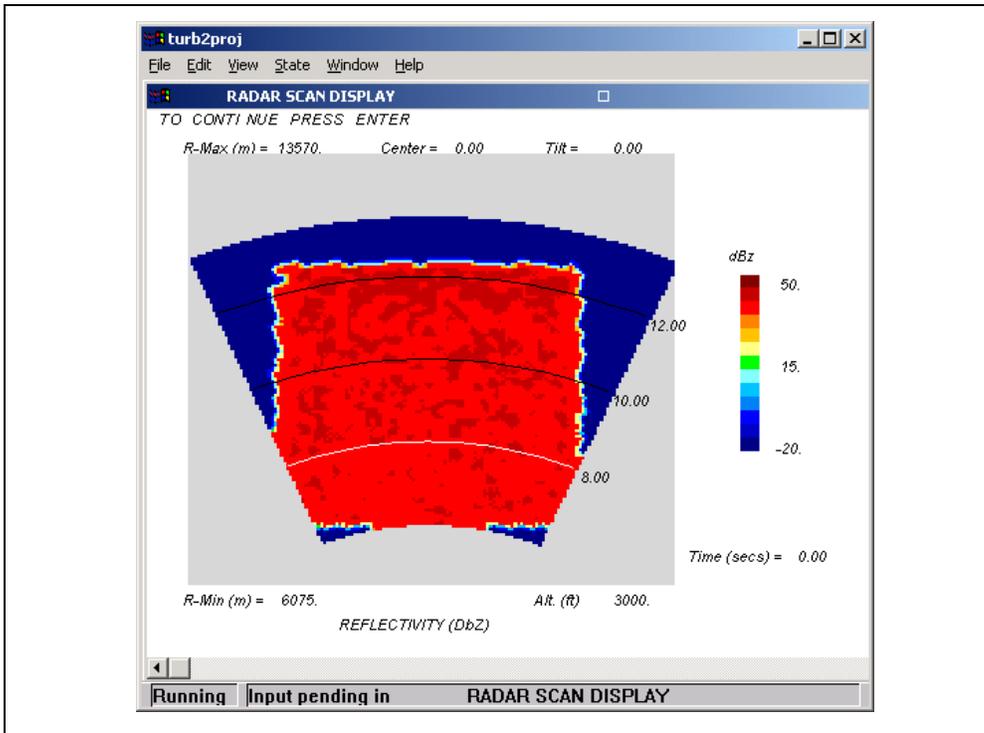


Figure 5 - Reflectivity Produced by ADWRS 4.0

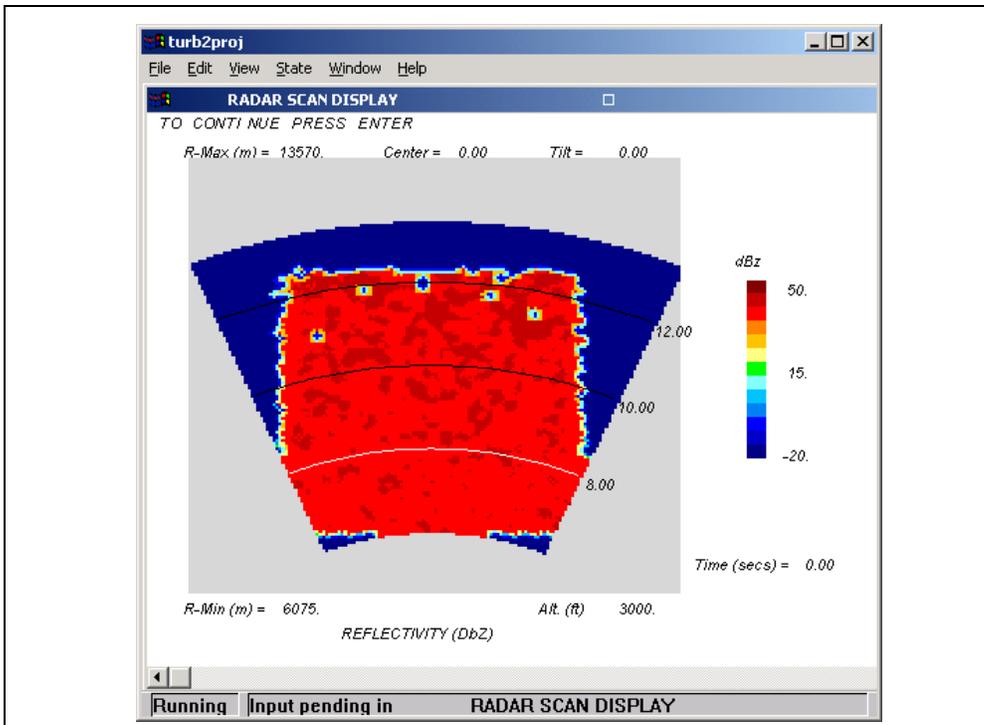


Figure 6 - Reflectivity Produced by ADWRS 5.0

Calculated ADWRS Reflectivity (dBZ)

TASS data run 191-06 (Time=48min, Alt=10.3km)

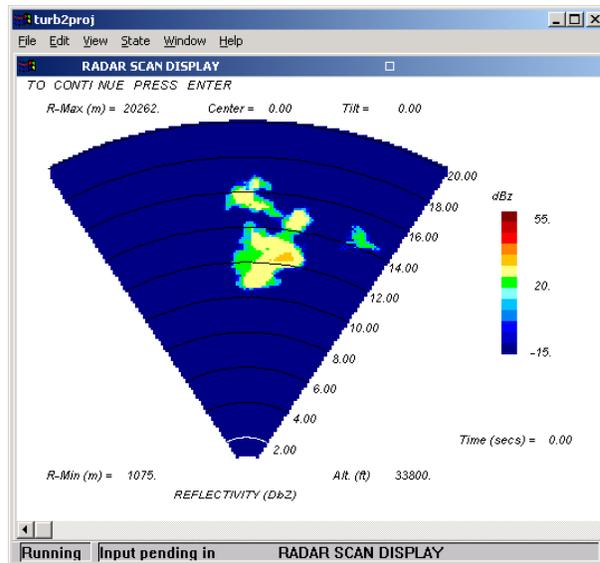


Figure 7 - Reflectivity of Tass Data Produced by ADWRS 5.0

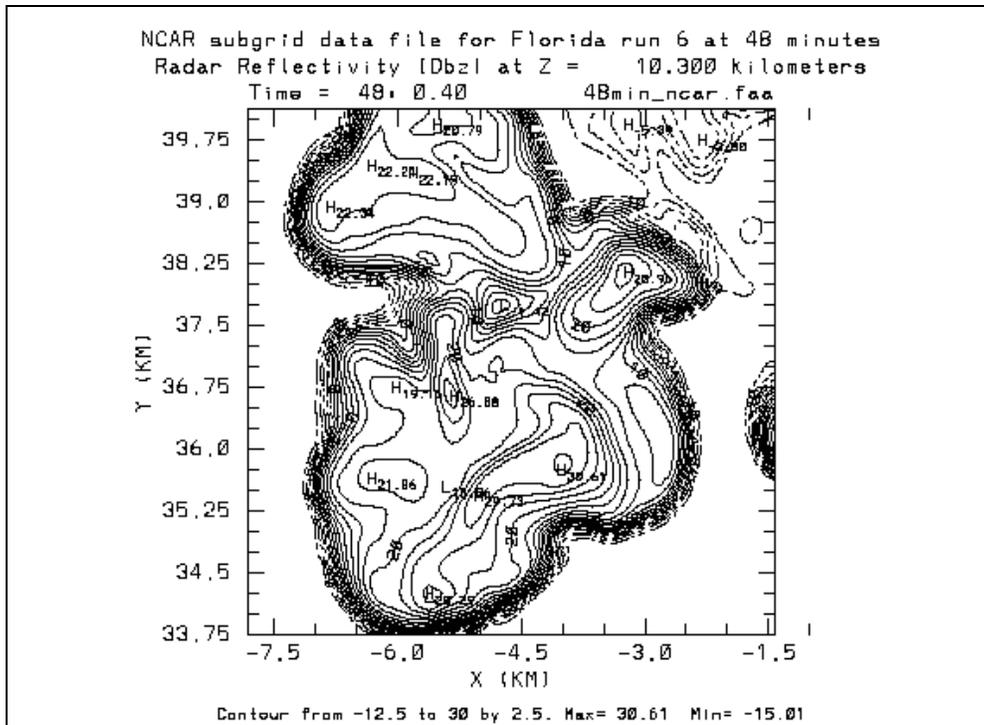


Figure 8 - Reflectivity of Tass Data Produced by TASS